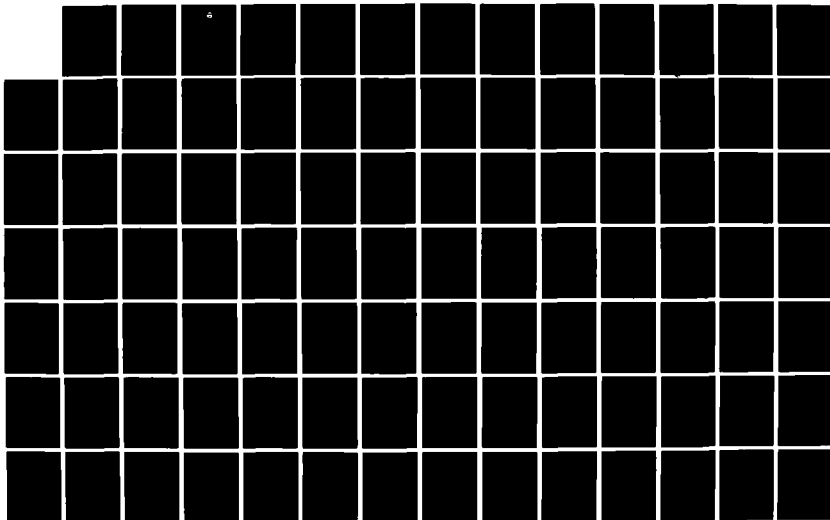
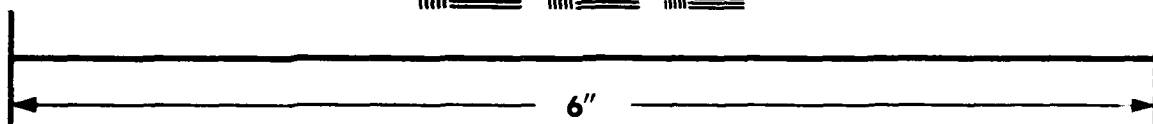
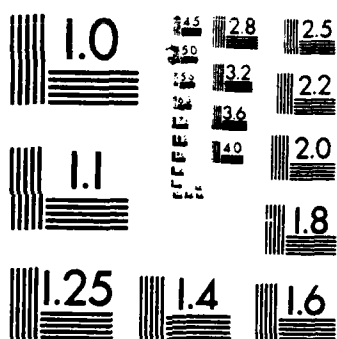
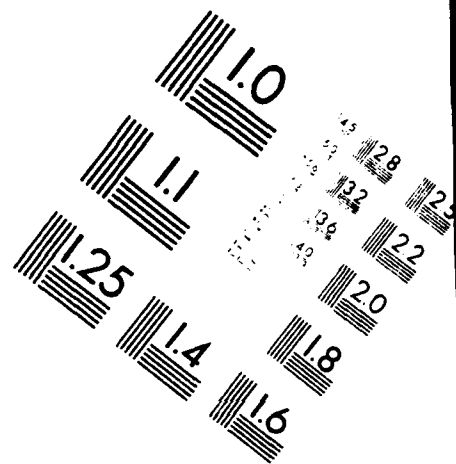
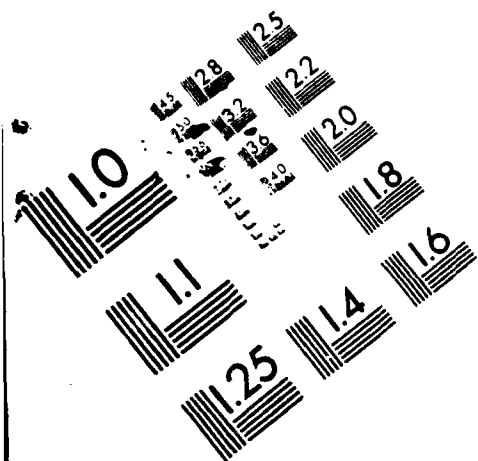


ADA 139626

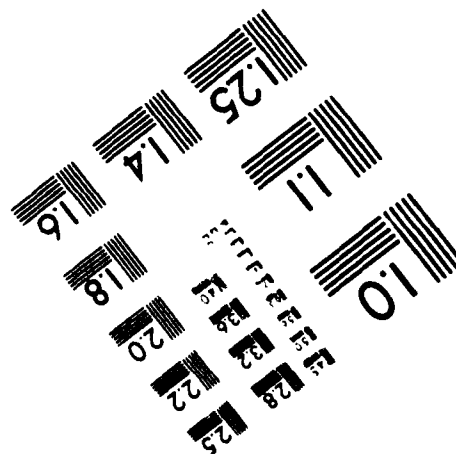
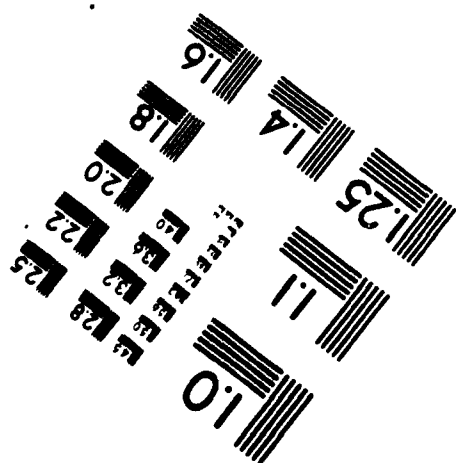
NAVAL OCEAN SYSTEMS CENTER, SAN DIEGO, CA
INGRES TO DBASE AND FINANCIAL SPREADSHEET
PROGRAM PACKAGES BY J BAIRD SYSTEM
DEVELOPMENT CORPORATION

1 OF 2
NOSC CR 223
UNCLASSIFIED
FEB 1984





MICROCOPY RESOLUTION TEST CHART



Contractor Report 223

**INGRES TO DBASE AND FINANCIAL
SPREADSHEET PROGRAM PACKAGES**

System Development Corporation
J. Baird (NOSC)
Task Coordinator

February 1984

Prepared for
Naval Ocean Systems Center
Code 912

Approved for public release; distribution unlimited.

NOSC

NAVAL OCEAN SYSTEMS CENTER
San Diego, California 92152



NAVAL OCEAN SYSTEMS CENTER SAN DIEGO, CA 92152

AN ACTIVITY OF THE NAVAL MATERIAL COMMAND

J.M. PATTON, CAPT, USN
Commander

R.M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

Administrative information pertaining to Naval Ocean Systems Center Contractor Report 223 is listed below.

<u>Item</u>	<u>Data</u>
Performing Organization	System Development Corporation 4065 Hancock Street San Diego, CA 92110
Contract Number	N66001-83-D-0094
Controlling Office	Naval Ocean Systems Center Code 912 San Diego, CA 92152
Funded by	NOSC, Project Management Support System (PMSS)
Released by J. A. Gilbreath, Head Computer Sciences and Simulation Department	Under authority of H. R. Talkington, Director Engineering and Computer Sciences Directorate

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NOSC CR 223	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) INGRES TO DBASE AND FINANCIAL SPREADSHEET PROGRAM PACKAGES		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) J. Baird (NOSC) Task Coordinator		8. CONTRACT OR GRANT NUMBER(s) N66001-83-D-0094
9. PERFORMING ORGANIZATION NAME AND ADDRESS System Development Corporation 4065 Hancock Street San Diego, CA 92110		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE February 1984
		13. NUMBER OF PAGES 109
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Naval Ocean Systems Center Code 912 San Diego, CA 92152		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Relational Data Base Ingres Translator Data Management Formats DBase Microcomputers		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The <u>ingresdbase</u> command provides support for exchanging user database files between two popular relational database management systems. <u>Ingres</u> (by Relational Technologies, Inc) and <u>dBASE II</u> (by Ashton-Tate, Inc) databases may be translated to or from their respective internal formats into a form suitable for transporting over standard ASCII data communication links. The command includes sufficient information to automatically re-create the database on the other machine with only modest user involvement.		

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)



S N 0102- LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

TABLE OF CONTENTS

		<u>Page</u>
SECTION 1	INTRODUCTION	
1.1	Purpose.....	1-1
1.2	Scope.....	1-1
 SECTION 2	 INGRES/DBASE TRANSLATION PROGRAMS	
2.1	Ingresdbase Command (man).....	2-1
2.2	Nroff Instruction File (man).....	2-3
2.3	Makefile (csh).....	2-5
2.4	Ingresdbase (csh).....	2-6
2.5	Ingresid.q (c).....	2-9
2.6	Ingresdi.c (c).....	2-20
2.7	Ingresul.c (c).....	2-40
2.8	Ingresdb.c (c).....	2-50
2.9	Ingres.cmd (d).....	2-64
2.10	Ingresmm.cmd (d).....	2-65
2.11	Ingresan.cmd (d).....	2-69
2.12	Ingrescd.cmd (d).....	2-71
2.13	Ingrescr.cmd (d).....	2-72
2.14	Ingresdr.cmd (d).....	2-74
2.15	Ingresed.cmd (d).....	2-75
2.16	Ingresix.cmd (d).....	2-77
2.17	Ingresrb.cmd (d).....	2-80
2.18	Ingresss.cmd (d).....	2-81
 SECTION 3	 SUPERCOMP-TWENTY PROGRAMS	
3.1	Makefile.sup (csh).....	3-1
3.2	Super Command (man).....	3-2
3.3	Super Nroff File (man).....	3-5
3.4	Superd Command (man).....	3-7
3.5	Superd Nroff File (man).....	3-9
3.6	Supert Command (man).....	3-11
3.7	Supert Nroff File (man).....	3-12
3.8	Supert.c (c).....	3-13

SECTION 1 INTRODUCTION

1.1 Purpose.

The purpose of this document is to describe the computer program packages and related equipments being delivered under NOSC Contract N66001-83-D-0094, Delivery Order 002203. The following materials will be divided between two major topics. The first topic is support for the Ingres-to-dBaseII translator, the second is support for the electronic spread sheet package, Supercomp-Twenty. These programs are part of the larger Project Management Support System (PMSS). PMSS has as one of its objectives to make a set of tools available throughout NOSC which are interoperable with the Center's central computers and with user's small microcomputer based systems. The programs being delivered in this document are in support of this objective.

1.2 Scope.

The scope of this document shall be limited to the source program listings for the programs being delivered, shell scripts or submit files, and documentation manual pages. The table of contents has been expanded to include the following notations as an aid to quickly identifying each component:

- (c) Indicates a C Language Program
- (d) Indicates a dBaseII Command Language Program
- (csh) Indicates a UNIX C-shell Script
- (sub) Indicates a CP/M Submit file
- (man) Indicates a Documentation Manual Page

*INGRES is a trademark of Relational Technologies, Inc.; DBASE II is a trademark of Ashton-Tate, Inc.; CP/M is a trademark of Digital Research, Inc.; SUPERCOMP-TWENTY is a trademark of Access Technology, Inc.; UNIX is a trademark of Bell Laboratories.

SECTION 2

SECTION 2.1

ingresdbase (1L) UNIX Programmer's Manual ingresdbase (1L)

NAME

ingresdbase - ingres / dBASE II translator

SYNOPSIS

ingresdbase <id|di><database><relation>

DESCRIPTION

The ingresdbase command provides support for exchanging user database files between two popular relational database management systems. Ingres (by Relational Technologies, Inc) and dBASE II (by Ashton-Tate, Inc) databases may be translated to or from their respective internal formats into a form suitable for transporting over standard ASCII data communication links. The command will include sufficient information to automatically re-create the database on the other machine with only modest user involvement.

Command line options are:

id requests ingres-to-dbase translation.

di requests dbase-to-ingres translation.

<database> specifies the ingres database to use. The specified database must exist. You must also have adequate permissions to create temporary relations used in the conversion process.

<relation> specifies the ingres relation to be used. When translating ingres-to-dbase the relation must exist and you must have permission to read it. When translating dbase-to-ingres the relation will be created if it does not already exist. If it exists, you must own the relation since it will be deleted and re-created.

The translation process will yeild a file in the form <relation>.s when using the ingres-to-dbase option. The translation process will require as input a file in the form of <relation>.s when using the the dbase-to-ingres option. You will be asked to "upload" or "download" this file from your microcomputer when appropriate.

The complete operation of this command requires the use of sumilar programs on the users remote microcomputer. At NOSOC, a program diskette is available from [supply complete reference] for exchanging ingres databases on CCVAX and PMSSVAX with dBASE II operating on CP/M and MS-DOS microcomputers.

FILES

ingresdbase will create a temporary sub-directory "tmp" to contain all working files needed by the translation process. This directory will be deleted at the end of the process.

DIAGNOSTICS

Various messages may be issued by ingres depending upon your permission and ownership status. Consult your ingres manual for additional explanation.

BUGS

This command assumes that the user is familiar with the operation of both INGRES and dBASE II.

The "upload" and "download" functions are internal to this command (using umodem). They will be modified when a PMSS standard file transfer program is established.

The present version of RTI ingres does not provide fixed decimal point notation used by dBASE II. This translator makes some "best guess" judgements when floating point numbers are included in the relation.

AUTHOR

Mike Trest, System Development Corporation
C/O John Baird, NOSC Code 912

SECTION 2.2

INGRES/DBASE NROFF FILE

```
.TH ingresdbase 1L NCSC
.UC 4
.SH NAME
ingresdbase \- ingres / dBASE II translator
.SH SYNOPSIS
.B ingresdbase
.RB <id|di> <database> <relation>
.SH DESCRIPTION
The
.I ingresdbase
command provides support for exchanging
user database files
between two popular relational database
management systems.
.I Ingres
(by Relational Technologies, Inc) and
.I dBASE II
(by Ashton-Tate, Inc) databases may be translated
to or from their respective
internal formats into a form suitable
for transporting over standard
ASCII data communication links.
The command will
include sufficient information to
automatically re-create the database
on the other machine with only modest
user involvement.
.sp
Command line options are:
.TP 1.2i
.I id
requests ingres-to-dbase translation.
.TP
.I di
requests dbase-to-ingres translation.
.TP
.I <database>
specifies the ingres database to use.
The specified database must exist. You
must also have adequate permissions to create temporary
relations used in the conversion process.
.TP
.I <relation>
specifies the ingres relation to be used.
When translating ingres-to-dbase the relation must exist
and you must have permission to read it. When translating
dbase-to-ingres the relation will be created if it does
not already exist. If it exists, you must
own the relation since it will be deleted and
re-created.
```

.SH
The translation process will yeild a file in the form
.I <relation>.s
when using the ingres-to-dbase option. The translation
process will require as input a file in the form of
.I <relation>.s
when using the the dbase-to-ingres option. You will be
asked to "upload" or "download" this file from your
microcomputer when appropriate.

.sp
The complete operation of this command
requires the use of sumilar programs
on the users remote microcomputer.
At NOSC, a program diskette is available
from
[supply complete reference]
for exchanging ingres databases on
CCVAX and PMSSVAX with dBASE II operating
on CP/M and MS-DOS microcomputers.

.SH FILES
.I ingresdbase
will create a temporary sub-directory "tmp" to contain all
working files needed by the translation process. This
directory will be deleted at the end of the process.

.SH DIAGNOSTICS
Various messages may be issued by
.I ingres
depending upon your permission and ownership status. Consult
your ingres manual for additional explanation.

.SH BUGS
This command assumes that the user is familiar with
the operation of both INGRES and dBASE II.

.sp
The "upload" and "download" functions are internal to
this command (using
.I umodem
). They will be modified when a PMSS
standard file transfer program is established.

.sp
The present version of RTI ingres does not provide fixed
decimal point notation used by dBASE II. This translator
makes some "best guess" judgements when floating point numbers are
included in the relation.

.SH AUTHOR
Mike Trest, System Development Corportation
.br
C/O John Baird, NOSC Code 912

SECTION 2.3

File: MAKEFILE Size: 1K

Page: 1

```
1: default: programs
2:
3: programs:
4:     @echo "The 'c' programs for the VAX "
5:     @echo "  ingresid      ingres-to-dbase translator"
6:     @echo "  ingresdi      dbase-to-ingres translator"
7:     @echo
8:     @echo "The 'c' programs for the micro"
9:     @echo "  ingresdb      ingres-to-dbase translator"
10:    @echo "  ingresul      dbase-to-ingres translator"
11:    @echo
12:    @echo "The 'dbase.cmd' programs for the micro"
13:    @echo "  INGRESMM      main menu program"
14:    @echo "  INGRESR      create relation"
15:    @echo "  INGRESR      delete relation"
16:    @echo "  INGRESR      clear data from relation"
17:    @echo "  INGRESR      edit data of relation"
18:    @echo "  INGRESR      switch to another relation"
19:    @echo "  INGRESR      show structure of relation"
20:    @echo "  INGRESR      build stream for vax ingres"
21:
22: ingresdi: ingresdi.c
23:     cc ingresdi.c -o ingresdi
24:
25: ingresid: ingresid.q
26:     equal ingresid.q
27:     cc ingresid.c \
28:         ~ingres/lib/libq.a ~ingres/lib/iutil ~ingres/lib/gutil \
29:         -o ingresid
30:     rm ingresid.c
```

Characters: 909

```

1: #csh
2: #
3: # INGRES/DBASE TRANSLATOR
4: #
5: set idv=1.2
6: set id=$1
7: set idd=$2
8: set idr=$3
9: set debug=$4
10: set idm=r_$idr
11: set admin='/etc/delivermail -eq -n v:trest '
12: set idup='/va/trest/bin/umodem -rt7l'
13: set iddown='/va/trest/bin/umodem -st7l'
14: clear
15: echo "ingresdbase: $id $idd $idr ..starting.."
16: if (" $user" != "trest") then
17:     echo "ingresdbase: INGRES/DBASE TRANSLATOR (version $idv)"
18:     echo
19:     echo "THIS IS AN EXPERIMENTAL PROGRAM FOR MOVING DATABASE"
20:     echo "APPLICATIONS FROM INGRES (ON CCVAX:) TO YOUR"
21:     echo "MICROCOMPUTER USING dBASE II."
22:     echo
23:     echo "Please mail comments, bugs, etc. to"
24:     echo "  To: ty"
25:     echo "  Subject: ingresdbase comments"
26:     echo
27: endif
28: if ((" $id" != "id") && (" $id" != "di")) || \
29:     ((" $idr" == "") || (" $idd" == "")) then
30:     echo " "
31:     echo -n ""
32:     echo "ingresdbase: INGRES/DBASE TRANSLATOR (version $idv)"
33:     echo " "
34:     echo "Usage: ingresdbase: <id|di> <database> <relation>"
35:     echo "where 'id' is ingres-to-dbase requested"
36:     echo "      'di' is dbase-to-ingres requested"
37:     echo "      <database> is the ingres database to use"
38:     echo "      <relation> is the ingres relation to use"
39:     echo " "
40:     echo "      If <relation>.s does not exist in the"
41:     echo "      current directory, umodem will be called"
42:     echo "      to upload it from the micro."
43:     echo " "
44:     exit
45: endif
46: set idp=`pwd`
47: set idw=$idp/tmp
48: if (" $idp" == " $idw") then
49:     echo "csh has failed. Please this failure to system support."
50:     exit

```

```
51: else
52:     rm -f -r $idw
53:     mkdir $idw
54:     echo "Subject: ingresdbase: usage $USER " >> $idw/usage
55:     echo "" >> $idw/usage
56:     echo -n `date` >> $idw/usage
57:     echo " ingresdbase: $id $idd $idr " >> $idw/usage
58: endif
59: #
60: # INGRES-TO-DBASE
61: #
62: if (" $id" = "id") then
63:     echo "Building $idw/$idr.s from INGRES: $idd $idr"
64:     ingresid $idd $idr $idw $debug
65:     if ( -e $idw/$idm.mo) then
66:         echo "destroy $idm \\g" > $idw/$idm.rm
67:         ingres -s $idd < $idw/$idm.rm
68:         mv -f $idw/$idm.s $idw/$idr.s
69:     endif
70:     echo -n "ready to download $idw/$idr.s now? (y/n)"
71:     set q=$<
72:     if ($q = "") set q="y"
73:     if (($q != "y") && ($q != "n")) set q="n"
74:     if ($q = "y") then
75:         echo "Ok: $iddown $idw/$idr.s"
76:         $iddown $idw/$idr.s
77:     else
78:         mv -f $idw/$idr.s $idp/$idr.s
79:         echo "No download. $idw/$idr.s saved in $idp"
80:     endif
81:     echo `date` >> $idw/usage
82:     $admin < $idw/usage
83:     if (" $debug" != "debug") then
84:         rm -r -f $idw
85:     endif
86:     echo "ingresdbase: $id $idd $idr ..completed.."
87:     exit
88: endif
89: #
90: # DBASE-TO-INGRES
91: #
92: if (" $id" = "di") then
93:     if (!( -e $idw/$idr.s) && ( -e $idp/$idr.s)) then
94:         echo "ingresdbase: USING EXISTING FILE $idp/$idr.s "
95:         ln $idp/$idr.s $idw
96:     endif
97:     if (!( -e $idw/$idr.s)) then
98:         echo "ingresdbase: $idr.s is being uploaded from micro"
99:         echo "ingresdbase: $idup $idw/$idr.s"
100:         $idup $idw/$idr.s
```

```
101:         endif
102:         ingresdi $idd $idr $idw $debug
103:         if (-e $idw/$idr.ing) then
104:             if (-e $idw/$idr.dat) then
105:                 echo "INGRES: $idd < $idw/$idr.ing"
106:                 ingres -s $idd < $idw/$idr.ing
107:             else
108:                 echo "ingresdbase: ERROR: $idw/$idr.dat not found"
109:                 exit
110:             endif
111:         else
112:             echo "ingresdbase: $idw/$idr.ing not present"
113:             echo "ingresdbase: trying $idw/$idm.ing"
114:             if (-e $idw/$idm.ing) then
115:                 if (-e $idw/$idm.mi) then
116:                     echo "ingresdbase: using $idw/$idm.ing"
117:                     echo "" >> $idw/$idm.ing
118:                     cat $idw/$idm.mi >> $idw/$idm.ing
119:                 endif
120:                 if (-e $idw/$idm.dat) then
121:                     echo "INGRES: $idd < $idw/$idm.ing"
122:                     ingres -s $idd < $idw/$idm.ing
123:                 else
124:                     echo "ingresdbase: ERROR: $idw/$idm.dat not found"
125:                     exit
126:                 endif
127:             else
128:                 echo "ingresdbase: ERROR: cannot find $idw/$idr.ing"
129:                 echo "ingresdbase:      or re-mapped $idw/$idm.ing"
130:                 exit
131:             endif
132:         endif
133:     endif
134:     echo `date` >> $idw/usage
135:     $admin < $idw/usage
136:     if (" $debug" != "debug") then
137:         rm -r -f $idw
138:     endif
139:     echo "ingresdbase: Sid $idd $idr ..completed.."
140:     exit
141: endif
```

Characters: 3846

```

1:  /*
2:  ****
3:  *      ingresid.q "Get INGRES attributes for database relation"
4:  *      build <relation>.s file for porting to micro
5:  ****
6:  *
7:  *      PROGRAM RUNS ON VAX 11/780 UNDER UNIX
8:  *
9:  *
10: *      Usage:  ingresid [-s|r] <database> <relation> <wdn>
11: *
12: *      <database> name of an existing database
13: *      <relation> name of a relation in that database
14: *      <wdn>      working directory name
15: ****
16: */
17:
18: #include <stdio.h>
19:
20: #define MAXATTR 32          /* maximum attributes per relation */
21: #define MAXNAME 10         /* maximum size of attribute name */
22:
23: /*
24: *      these items must be known to EQUEL, INGRES and C
25: */
26: ##      char      *database;
27: ##      char      *relation;
28: ##      char      *datafile;
29: ##      char      relid[20];
30: ##      char      name[20];
31: ##      char      frmt[2];
32: ##      int       id;
33: ##      int       frml;
34: ##      int       off;
35:
36: /*
37: *      following items known only to C
38: */
39: struct attrib                /* array of attribute descriptions */
40: {
41:     char name[MAXNAME+1];    /*attribute name*/
42:     char frmt;               /*attribute format type*/
43:     int  frml;               /*attribute format length*/
44:     int  decimal;            /*attribute decimal positions*/
45:     int  offset;             /*attribute starting location*/
46: } attr[MAXATTR], *attrcur, orig[MAXATTR], *origcur;
47:
48: FILE      *stream;
49:
50: char      *ptr1, *ptr2, *ptr3;

```

```
51:
52: char    wdn[120];
53: char    file_s[120];
54: char    file_dat[120];
55: char    re_mapped[120];
56: char    map_out[120];
57: char    map_in[120];
58: char    map_temp[120];
59: char    command[120];
60:
61: int      i,ii;
62: int      current,last,width,nonascii;
63: int      flagr,flags;
64:
65: /*
66:  *****
67:  *      this is a main program which expects command line arguments
68:  *****
69:  */
70: main(argc,argv)
71: int      argc;
72: char    *argv[];
73: {
74:
75:     while(argc > 1 && argv[1][0] == '-')
76:     {
77:         for(i=1;argv[1][i];i++)
78:         {
79:             switch(argv[1][i])
80:             {
81:             case 'r':
82:             case 'R': flagr++; break;
83:             case 's':
84:             case 'S': flags++; break;
85:             }
86:         }
87:         argc--;
88:         argv++;
89:     }
90:     if (!flagr && !flags) flagr=flags=1;
91:     if (argc <= 3)
92:     {
93:         printf("Usage: ingresid <database> <relation> <wdn>\n");
94:         printf("where          <database> name of an existing database\n");
95:         printf("          <relation> name of a relation in database\n");
96:         printf("          <wdn>      working directory name");
97:         exit(-1);
98:     }
99:     database = argv[1];
100:    relation = argv[2];
```

```
101:      datafile = file_dat;
102:      strcpy(wdn,argv[3]);
103:      strcat(wdn,"/");
104:
105:
106:      collect();          /* get attributes */
107:      if (current)        /* if any attributes found */
108:      {
109:          if (nonascii && flagr)
110:          {
111:              re_map();    /* change the relation */
112:              collect();   /* get new attributes */
113:          }
114:          if (flags)
115:          {
116:              strcpy(file_s,wdn);
117:              strcat(file_s,relation);
118:              strcat(file_s,".s");
119:              strcpy(file_dat,wdn);
120:              strcat(file_dat,relation);
121:              strcat(file_dat,".dat");
122:              if ((stream = fopen(file_s,"w+")) == NULL)
123:              {
124:                  printf("->ingresid unable to generate %s\n",
125:                          file_s);
126:                  exit(-1);
127:              }
128:              mapper();
129:              ingres();
130:              dbase();
131:              tdata();
132:              fclose(stream);
133:          }
134:          exit(1);
135:      }
136:      else    exit(0);
137: }
138:
139: /*
140: *****
141: * collect attribute information into array "attr"
142: *****
143: */
144: collect()
145: {
146:     printf("->ingresid: Collecting attribute information about %s\n",
147:             relation);
148:     last = current = width = nonascii = 0;
149:     ##      ingres database
150:     ##      range of a is attribute
```

```

151: ##      retrieve (relid=a.attrelid,
152: ##                  name=a.attname,
153: ##                  id=a.attid,
154: ##                  off=a.attoff,
155: ##                  frmt=a.attfrmt,
156: ##                  frml=a.attfrml)
157: ##      where
158: ##          a.attrelid = relation
159: ##      {
160:      current++;
161:      last = current;
162:      attrcur = &attr[current];
163:      if(*name)
164:      {
165:          while(name[(strlen(name)-1)] == ' ')
166:              name[(strlen(name)-1)] = '\0';
167:      }
168:      name[MAXNAME] = '\0';
169:      strcpy(attrcur->name,name);
170:      attrcur->frmt = *frmt;
171:      attrcur->frml = frml;
172:      attrcur->offset = width;
173:      attrcur->decimal = 0;
174:      width += frml;
175:      if (*frmt != 'c') nonascii++;
176: ##      }
177: ##      exit
178:      if (last>31)
179:      {
180:          printf("\007\n->ingresid: Caution %s has ", relation);
181:          printf("%d fields.\n", last);
182:          printf("->ingresid: This exceeds dBASEII limits!\007\n");
183:          exit(-1);
184:      }
185: }
186:
187: /*
188: *****
189: * build a dBASEII definition
190: *****
191: */
192: dbase()
193: {
194:     printf("->ingresid: Moving dBASEII definition to %s\n",
195:            file_s);
196:     fprintf(stream, ".dbase %s %d \n",relation,width);
197:     for(current = 1;current <= last; current++)
198:     {
199:         attrcur = &attr[current];
200:         origcur = &orig[current];

```

```

201:         fprintf(stream,"%-10s",attrcur->name);
202:         if (origcur->fmt == 'i' ||
203:             origcur->fmt == 'f')
204:             fprintf(stream,"n");
205:         else fprintf(stream,"%lc",attrcur->fmt);
206:         fprintf(stream,"%3.3d",attrcur->frml);
207:         if (origcur->fmt == 'f')
208:             fprintf(stream,"003");
209:         else fprintf(stream,"%3.3d",attrcur->decimal);
210:         if (current == last)
211:             fprintf(stream,"\n");
212:         else fprintf(stream,"\n");
213:     }
214:     fprintf(stream,".end\n");
215: }
216:
217: /*
218:  *****
219:  * build a ingres create/copy statement
220:  *****
221:  */
222: ingres()
223: {
224:     printf("->ingresid: Moving INGRES definition to %s\n",
225:         file s);
226:     fprintf(stream,".ingres %s %d \n",relation,width);
227:     fprintf(stream,"destroy %s \\g\n",relation);
228:     fprintf(stream,"create %s (\n",relation);
229:     for(current = 1;current <= last; current++)
230:     {
231:         attrcur = &attr[current];
232:         fprintf(stream,"      %s=%c%d",
233:             attrcur->name,
234:             attrcur->fmt,
235:             attrcur->frml);
236:         if (current == last)
237:             fprintf(stream,"\\)\n");
238:         else fprintf(stream,",\n");
239:     }
240:     fprintf(stream,"\\g\n");
241:
242:     fprintf(stream,"copy %s (\n",relation);
243:     for(current = 1;current <= last; current++)
244:     {
245:         attrcur = &attr[current];
246:         fprintf(stream,"      %s=%c%d",
247:             attrcur->name,
248:             attrcur->fmt,
249:             attrcur->frml);
250:         if (current == last)

```

```

251:                fprintf(stream, "\) from \"%s\"\\n", datafile);
252:            else    fprintf(stream, ",\\n");
253:        }
254:        fprintf(stream, "\\g\\n");
255:        fprintf(stream, ".end\\n");
256:    }
257:
258: /*
259:  *****
260:  * output entire table in bulk form to a file
261:  *****
262:  */
263: tdata()
264: {
265:     FILE *fin;
266:     int i, ii, d, o;
267:     char sign;
268:     char fbuffer[BUFSIZ*8];
269:
270:     printf("->ingresid: Moving INGRES data to %s\\n",
271:            file_s);
272: ##    ingres database
273: ##    copy    relation () into datafile
274: ##    exit
275:     if ((fin = fopen(datafile, "r")) == NULL)
276:     {
277:         printf("\\007->ingresid: Can't Read from %s\\n", datafile);
278:         exit(-1);
279:     }
280:     fprintf(stream, ".data %s %d \\n", relation, width);
281:     for( ; (ii = fread(fbuffer, 1, width , fin)) > 0; )
282:     {
283:         fbuffer[ii] = '\\0';
284:         /* right adjust & zero fill int and float fields */
285:         for(current = 1; current <= last; current++)
286:         {
287:             origcur = &orig[current];
288:             if (origcur->frmt == 'i' ||
289:                 origcur->frmt == 'f')
290:             {
291:                 attrcur = &attr[current];
292:                 i=attrcur->frml;
293:                 d=attrcur->frml;
294:                 o=attrcur->offset;
295:                 ptr1 = &fbuffer[o];
296:                 ptr2 = &fbuffer[o + (i-1)];
297:                 ptr3 = ptr2;
298:                 sign = '\\0';
299:                 if (*ptr1 == '-')
300:                 {

```

```

301:             sign = *ptr1;
302:             *ptr1 = '0';
303:         }
304:         while(i && *ptr2 == ' ')
305:         {
306:             ptr2-- ;
307:             i-- ;
308:             d = i ;
309:         }
310:
311:         while(i)
312:         {
313:             if (*ptr2 == '.')
314:             {
315:                 attrcur->decimal = d-i ;
316:                 *ptr2-- ;
317:             }
318:             else *ptr3-- = *ptr2-- ;
319:             i-- ;
320:         }
321:         while(ptr3 >= ptr1)
322:         {
323:             *ptr3-- = '0';
324:         }
325:         if (sign) *ptr1 = sign;
326:     }
327: }
328:     fwrite(fbuffer, 1, ii, stream) ;
329: }
330: fclose(fin);
331: unlink(datafile);
332: fprintf(stream, "\n.end\n");
333: }
334:
335: /*
336: *****
337: * output re map instruction files, if present
338: * execute the ingres ".mo" instructions
339: * change working name of relation
340: *****
341: */
342: mapper ()
343: {
344:     FILE *fin;
345:     int ii;
346:     char fbuffer[BUFSIZ];
347:     char fname[120];
348:
349:     printf("->ingresid: Moving INGRES re_mapping information to %s\n",
350:           file_s);

```

```
351:      /* copy the output mapping instructions */
352:      strcpy(fname,wdn);
353:      strcat(fname,relation);
354:      strcat(fname,".mo");
355:      if ((fin = fopen(fname,"r")) == NULL)
356:          return(-1);
357:      fprintf(stream,".mapper %s %s.mo \n",relation,relation);
358:      for( ; (ii = fread(fbuffer, 1, BUFSIZ, fin)) > 0; )
359:          {
360:              fwrite(fbuffer, 1, ii, stream) ;
361:          }
362:      fclose(fin);
363:      fprintf(stream,".end\n");
364:      /* copy the input mapping instructions */
365:      strcpy(fname,wdn);
366:      strcat(fname,relation);
367:      strcat(fname,".mi");
368:      if ((fin = fopen(fname,"r")) == NULL)
369:          return(-1);
370:      fprintf(stream,".mapper %s %s.mi \n",relation,relation);
371:      for( ; (ii = fread(fbuffer, 1, BUFSIZ, fin)) > 0; )
372:          {
373:              fwrite(fbuffer, 1, ii, stream) ;
374:          }
375:      fclose(fin);
376:      fprintf(stream,".end\n");
377:      return(1);
378: }
379:
380: /*
381: *****
382: * build ingres re-map files
383: *****
384: */
385: re_map()
386: {
387:
388:     strcpy(map_temp,"r ");
389:     strcat(map_temp,relation);
390:     strcpy(map_out,wdn);
391:     strcat(map_out,map_temp);
392:     strcat(map_out,".mo");
393:     strcpy(map_in,wdn);
394:     strcat(map_in,map_temp);
395:     strcat(map_in,".mi");
396:
397:     if ((stream = fopen(map_out,"w+")) == NULL)
398:     {
399:         printf("\007->ingresid: Unable to generate %s\n",map_out);
400:         return(-1);
```

```
401:     }
402:     printf("->ingresid: Re-mapping %s into %s\n",
403:            relation,map_temp);
404:     fprintf(stream,"destroy %s \\g\n",map_temp);
405:     fprintf(stream,"range of %s is %s \n",relation,relation);
406:     fprintf(stream,"retrieve into %s \\(\n",map_temp);
407:     for(current = 1;current <= last; current++)
408:     {
409:         attrcur = &attr[current];
410:         if (attrcur->fmt == 'c')
411:         {
412:             fprintf(stream,"          %s=%s.%s",
413:                    attrcur->name,
414:                    relation,
415:                    attrcur->name);
416:         }
417:         else
418:         {
419:             fprintf(stream,"          %s=ascii\\(%s.%s\\)",
420:                    attrcur->name,
421:                    relation,
422:                    attrcur->name);
423:         }
424:         if (current == last)
425:             fprintf(stream,"\\) \n");
426:         else
427:             fprintf(stream," , \n");
428:     }
429:     fprintf(stream,"\\g\n");
430:     fclose(stream);
431:     if ((stream = fopen(map_in,"w+")) == NULL)
432:     {
433:         printf("\007->ingresid Unable to generate %s\n",map_in);
434:         return(-1);
435:     }
436:     fprintf(stream,"destroy %s \\g\n",relation);
437:     fprintf(stream,"range of %s is %s \n",map_temp,map_temp);
438:     fprintf(stream,"retrieve into %s \\(\n",relation);
439:     for(current = 1;current <= last; current++)
440:     {
441:         attrcur = &attr[current];
442:         if (attrcur->fmt == 'c')
443:         {
444:             fprintf(stream,"          %s=%s.%s",
445:                    attrcur->name,
446:                    map_temp,
447:                    attrcur->name);
448:         }
449:         else
450:         {
```

```

451:         if (attrcur->frmt == 'i')
452:             fprintf(stream, "          %s=int%d\\(%s.%s\\)",
453:                 attrcur->name,
454:                 attrcur->frml,
455:                 map_temp,
456:                 attrcur->name);
457:         if (attrcur->frmt == 'f')
458:             fprintf(stream, "          %s=float%d\\(%s.%s\\)",
459:                 attrcur->name,
460:                 attrcur->frml,
461:                 map_temp,
462:                 attrcur->name);
463:     }
464:     if (current == last)
465:         fprintf(stream, "\\ \n");
466:     else
467:         fprintf(stream, ", \n");
468: }
469: fprintf(stream, "destroy %s \n", map_temp);
470: fprintf(stream, "\\g\n");
471: fclose(stream);
472: /* execute ".mo" ingres instructions */
473: strcpy(command, "ingres -s ");
474: strcat(command, database);
475: strcat(command, " < ");
476: strcat(command, map_out);
477: if (!fork ())
478: {
479:     execl ("/bin/sh", "-", "-c", command, 0);
480:     printf("\n\007->ingresid Cannot Execute: %s\n", command);
481:     return(-1);
482: }
483: wait(0);
484:
485: /* change working name for re-mapped relation */
486: /* and save a copy of the original attributes */
487: relation = map_temp;
488: for(current = 1; current <= last; current++)
489: {
490:     attrcur = &attr[current];
491:     origcur = &orig[current];
492:     strcpy(origcur->name, attrcur->name);
493:     origcur->frmt = attrcur->frmt;
494:     origcur->frml = attrcur->frml;
495:     origcur->offset = attrcur->offset;
496: }
497: }
498: /*
499: *****
500: * END OF SOURCE

```

File: INGRESID.Q

Size: 12K

Page: 11

501: *****

502: */

Characters: 12155

```

1: /*
2: *****
3: *      ingresdi.c break <relation>.s into its component files
4: *****
5: *
6: *      PROGRAM RUNS ON VAX-11/780
7: *
8: *      The input file is:
9: *          -> <relation>.s      stream coming from micro
10: *
11: *      The files created are:
12: *
13: *          1: <database>.DEF dBASEIII create definitions
14: *          2: <database>.DAT actual data file to be used
15: *          3: <database>.ING ingres create/copy command
16: *          4: <database>.MI ingres mapper instructions
17: *          note: <database>.ING and <database>.MI will
18: *              be generated by this program if not present
19: *              in the input <relation>.S file.
20: *
21: *
22: * SAMPLE DATA FILE INPUT STREAM
23: *
24: * NOTE: ALL LINES ARE <\n> delimited. This is very important
25: *      in the ".data" section. The data section will be checked
26: *      for fully padded records of <size=nnn> !PLUS ONE NEW LINE!
27: *      THE ".data" SECTION MUST BE THE LAST SECTION IN THE STREAM.
28: *
29:
30: .ingres clist 30
31: create clist (
32:     code=c5,
33:     name=c15,
34:     bldg=c5,
35:     room=c5)
36: copy clist (
37:     code=c5,
38:     name=c15,
39:     bldg=c5,
40:     room=c5) from "clist.dat"
41: .end
42: .dbase clist 30
43: code      c 5 0
44: name      c 15 0
45: bldg      c 5 0
46: room      c 5 0
47: .end
48: .data clist 30
49: 8241 Miyashiro      33    0047 <\n>
50: 8241 Woods, Bev     33    0067 <\n>

```

```
51: 8254 Campbell      33  0055 <\n>
52: 8241 Landa        33  0065 <\n>
53: .end
54:
55: *
56: ****
57: */
58: #include <stdio.h>
59: #include <ctype.h>
60:
61: #define BUFSIZE 512
62: #define BYTES_PER_BLOCK 128
63: #define ERROR -1
64: #define MAXNAME 10
65: #define MAXATTR 32
66:
67: struct attrib
68: {
69:     char name[MAXNAME+1];
70:     char frmt;
71:     int frml;
72:     int decimal;
73:     int offset;
74: } attr[MAXATTR], *attrcur, orig[MAXATTR], *origcur;
75:
76: int current, last, width, nonascii;
77: int s_mapper, s_dbase, s_data, s_ingres;
78: int filein, fileout;
79: int n, n2;
80: int bytes_read, bytes_written;
81: int rec_size, rec_bytes;
82: int signal_eof;
83:
84: FILE *stream, *fdat, *fwrk;
85:
86: char rel[20];
87: char file_s[120];
88: char file_map[120];
89: char file_ing[120];
90: char file_def[120];
91: char file_dat[120];
92: char file_wrk[120];
93: char wdn[120];
94: char work[120];
95:
96: char buffer[BUFSIZE+10];
97: char buffer2[BYTES_PER_BLOCK];
98:
99: char c;
100: char *ptr;          /* input stream buffer pointer */
```

```

101: char *ptr2;          /* output file buffer pointer */
102: char *database;       /* pointer to the database name */
103: char *relation;       /* pointer to the relation name */
104: char *indexer;        /* pointer to the index field name */
105:
106: char getabyte();
107: char ungetbyte();
108:
109: /*
110:  *****
111:  *      main control section is a loop which scans the input stream
112:  *      for ".mapper", ".dbase", ".ingres", or ".data" section headings.
113:  *      For each section an appropriate call is made to
114:  *      "do_thedata()" to extract the data for this section.
115:  *      At the end of all input gen_ing() is called if no ".ingres"
116:  *      section was read. un_map() is called if a ".mapper" section
117:  *      was read.
118:  *****
119:  */
120: main (argc,argv)
121: int  argc;
122: char *argv[];
123: {
124:
125:     if (argc < 4)
126:     {
127:         printf("Usage: ingresdi <database> <relation> <wdn>\n");
128:         printf("where          <database> name of an existing database\n");
129:         printf("          <relation> name of a relation in database\n");
130:         printf("          <wdn>      working directory name\n");
131:         exit(ERROR);
132:     }
133:     database = argv[1];
134:     relation = argv[2];
135:     strcpy (rel,argv[2]);
136:     strcpy (wdn,argv[3]);
137:     strcat (wdn,"/");
138:
139:     indexer = "NAME";
140:
141:     setup();
142:     while((c=getabyte()) != EOF)
143:     {
144:         if (c == '.')
145:         {
146:             ptr2 = work;
147:             *ptr2 = '\0';
148:             n2 = 25;
149:             while(n2-- && (c=getabyte()) &&
150:                 c != EOF &&

```

```

151:                                c != '\n' &&
152:                                c != ' ')
153:                                {
154:                                    *ptr2++ = c;
155:                                    *ptr2 = '\0';
156:                                }
157:                                if (!strcmp(work, "mapper"))
158:                                {
159:                                    do_mapper();
160:                                }
161:
162:                                if (!strcmp(work, "dbase"))
163:                                {
164:                                    s_dbase++;
165:                                    do_dbase();
166:                                }
167:
168:                                if (!strcmp(work, "ingres"))
169:                                {
170:                                    s_ingres++;
171:                                    do_ingres();
172:                                }
173:
174:                                if (!strcmp(work, "data"))
175:                                {
176:                                    s_data++;
177:                                    do_data();
178:                                }
179:                                } /*end of *ptr = '.' */
180:                                } /*end while getabyte */
181:                                collect();
182:                                if (s_dbase) gen_ing();
183:                                if (nonascii) gen_mi();
184:                                if (s_data && s_mapper) un_map();
185:                                printf("->ingresdi ok\n");
186:                                }
187:
188:                                /*
189:                                *****
190:                                *      setup      setup file name, open and prime input stream
191:                                *****
192:                                */
193:                                setup()
194:                                {
195:                                    strcpy(file_s, wdn);
196:                                    strcat(file_s, relation);
197:                                    strcat(file_s, ".s");
198:
199:                                    if ((filein = open(file_s, 0)) == ERROR)
200:                                    {

```

```

201:             printf("\n->ingresdi Can't Find %s\007\n",file_s);
202:             exit(ERROR);
203:         }
204:         else
205:         {
206:             n=read(filein,buffer+l0,BUFSIZE);
207:             bytes_read = n;
208:             ptr = buffer+l0;
209:         }
210:     }
211: /*
212: *****
213: *      do mapper      set up for mapper files from input stream
214: *****
215: */
216: do_mapper()
217: {
218:     char fil[60], *ex;
219:     char c;
220:
221:     /* flush white space */
222:     while((c=getabyte()) && c != EOF && c == ' ');
223:     ungetbyte(c);
224:     if (c == EOF) return;
225:     /* extract relation name */
226:     ex = fil;
227:     while((c=getabyte()) && c != EOF && c != '\n' && c != ' ')
228:     {
229:         if (isupper(c))
230:             *ex++ = tolower(c);
231:         else
232:             *ex++ = c;
233:     }
234:     *ex = '\0';
235:     strcpy(rel,fil);
236:     if (c == EOF) return;
237:     /* flush white space */
238:     while((c=getabyte()) && c != EOF && c == ' ');
239:     ungetbyte(c);
240:     if (c == EOF) return;
241:     /* extract file name */
242:     ex = fil;
243:     while((c=getabyte()) && c != EOF && c != '\n' && c != ' ')
244:     {
245:         if (isupper(c))
246:             *ex++ = tolower(c);
247:         else
248:             *ex++ = c;
249:     }
250:     *ex = '\0';
251:     if (c == EOF) return;
252:     /* flush single occurrence of <space><nl> */

```

```
251:         if ((c=getabyte()) && c != ' ')
252:             ungetbyte(c);
253:         if ((c=getabyte()) && !(c == '\n' || c == '\r'))
254:             ungetbyte(c);
255:         if ((c=getabyte()) && !(c == '\n' || c == '\r'))
256:             ungetbyte(c);
257:         if (c == EOF) return;
258:         /* create & copy data to the requested file*/
259:         strcpy(file_map,wdn);
260:         strcat(file_map,fil);
261:         printf("->ingresdi building %s\n",file_map);
262:         do_thedata(file_map,0);
263:     }
264:
265: /*
266:  *****
267:  *      do ingres      set up for ingres files from input stream
268:  *****
269:  */
270: do_ingres()
271: {
272:     char fil[60], *ex;
273:     char c;
274:
275:     /* flush white space */
276:     while((c=getabyte()) && c != EOF && c == ' ');
277:     ungetbyte(c);
278:     if (c == EOF) return;
279:     /* extract relation name */
280:     ex = fil;
281:     while((c=getabyte()) && c != EOF && c != '\n' && c != ' ')
282:     {
283:         if (isupper(c))
284:             *ex++ = tolower(c);
285:         else
286:             *ex++ = c ;
287:     }
288:     *ex = '\0';
289:     strcpy(rel,fil);
290:     if (c == EOF) return;
291:     /* flush white space */
292:     while((c=getabyte()) && c != EOF && c == ' ');
293:     ungetbyte(c);
294:     if (c == EOF) return;
295:     /* extract file size*/
296:     ex = fil;
297:     while((c=getabyte()) && c != EOF && c != '\n' && c != ' ')
298:     {
299:         if (isupper(c))
300:             *ex++ = tolower(c);
301:         else
302:             *ex++ = c ;
```

```

301:     }
302:     *ex = '\0';
303:     if (c == EOF) return;
304:     rec_size = atoi(fil);
305:     /* flush single occurrence of <space><nl> */
306:     if ((c=getabyte()) && c != ' ')
307:         ungetbyte(c);
308:     if ((c=getabyte()) && !(c == '\n' || c == '\r'))
309:         ungetbyte(c);
310:     if ((c=getabyte()) && !(c == '\n' || c == '\r'))
311:         ungetbyte(c);
312:     if (c == EOF) return;
313:     /* create & copy data to the requested file*/
314:     strcpy(file_ing,wdn);
315:     strcat(file_ing,rel);
316:     strcat(file_ing,".ing");
317:     printf("->ingresdi building %s\n",file_ing);
318:     do_thedata(file_ing,0);
319: }
320:
321: /*
322: *****
323: *      do dbase      set up for dbase files from input stream
324: *****
325: */
326: do_dbase()
327: {
328:     char fil[60], *ex;
329:     char c;
330:
331:     /* flush white space */
332:     while((c=getabyte()) && c != EOF && c == ' ');
333:     ungetbyte(c);
334:     if (c == EOF) return;
335:     /* extract relation name */
336:     ex = fil;
337:     while((c=getabyte()) && c != EOF && c != '\n' && c != ' ')
338:     {
339:         if (isupper(c))
340:             *ex++ = tolower(c);
341:         else
342:             *ex++ = c;
343:     }
344:     *ex = '\0';
345:     strcpy(rel,fil);
346:     if (c == EOF) return;
347:     /* flush white space */
348:     while((c=getabyte()) && c != EOF && c == ' ');
349:     ungetbyte(c);
350:     if (c == EOF) return;
351:     /* extract data relation size */

```

```
351:     ex = fil;
352:     while((c=getabyte()) && c != EOF && c != '\n' && c != ' ')
353:     {
354:         if (isupper(c))
355:             *ex++ = tolower(c);
356:         else *ex++ = c ;
357:     }
358:     *ex = '\0';
359:     if (c == EOF) return;
360:     rec_size = atoi(fil);
361:     /* flush single occurrence of <space><nl> */
362:     if ((c=getabyte()) && c != ' ')
363:         ungetbyte(c);
364:     if ((c=getabyte()) && !(c == '\n' || c == '\r'))
365:         ungetbyte(c);
366:     if ((c=getabyte()) && !(c == '\n' || c == '\r'))
367:         ungetbyte(c);
368:     if (c == EOF) return;
369:     /* create & copy data to the requested file*/
370:     strcpy(file_def,wcn);
371:     strcat(file_def,rel);
372:     strcat(file_def,".def");
373:     printf("->ingresdi building %s\n",file_def);
374:     do_thedata(file_def,0);
375: }
376:
377: /*
378: *****
379: *      do data      set up for data files from input stream
380: *****
381: */
382: do_data()
383: {
384:     char fil[60], *ex;
385:     char c;
386:
387:     /* flush white space */
388:     while((c=getabyte()) && c != EOF && c == ' ') ;
389:     ungetbyte(c);
390:     if (c == EOF) return;
391:     /* extract relation name */
392:     ex = fil;
393:     while((c=getabyte()) && c != EOF && c != '\n' && c != ' ')
394:     {
395:         if (isupper(c))
396:             *ex++ = tolower(c);
397:         else *ex++ = c ;
398:     }
399:     *ex = '\0';
400:     strcpy(rel,fil);
```

```

401:         if (c == EOF) return;
402:         /* flush white space */
403:         while((c=getabyte()) && c != EOF && c == ' ');
404:         ungetbyte(c);
405:         if (c == EOF) return;
406:         /* extract data relation size */
407:         ex = fil;
408:         while((c=getabyte()) && c != EOF && c != '\n' && c != ' ')
409:         {
410:             if (isupper(c))
411:                 *ex++ = tolower(c);
412:             else *ex++ = c;
413:         }
414:         *ex = '\0';
415:         if (c == EOF) return;
416:         rec_size = atoi(fil);
417:         /* Flush single occurrence of <space><nl> */
418:         if ((c=getabyte()) && c != ' ')
419:             ungetbyte(c);
420:         if ((c=getabyte()) && !(c == '\n' || c == '\r'))
421:             ungetbyte(c);
422:         if ((c=getabyte()) && !(c == '\n' || c == '\r'))
423:             ungetbyte(c);
424:         if (c == EOF) return;
425:         /* create & copy data to the requested file*/
426:         strcpy(file_dat,wcn);
427:         strcat(file_dat,rel);
428:         strcat(file_dat,".dat");
429:         printf("->ingresdi building %s [%d]\n",file_dat,rec_size);
430:         do_thedata(file_dat,1);
431:     }
432: /*
433: *****
434: *      do_thedata   writes current input stream to output buffer
435: *
436: *      NOTE: this routine is based upon this format:
437: *      see sample at front of this listing.
438: *
439: *      ".ingres<space>name<space>size-nnn<space><cr,lf>"
440: *      "<a stream of actual data (of record size-nnn)>"
441: *      "<cr,lf>.end<cr,lf>"
442: *
443: *      ALL DATA AFTER THE '.ingres' LINE AND UPTO THE '.end'
444: *      WILL BE WRITTEN TO THE OUTPUT FILE NAMED.
445: *
446: *      If 'datasw' is true, a standard CP/M <cr,lf> will be deleted
447: *      after every 'rec_size' bytes of data read in the ".data"
448: *      section.
449: *
450: *****

```

```
451: */
452: int
453: do_thedata(filename,datasw)
454: char *filename;
455: int datasw;
456: {
457:     char c;
458:     int fd, n2;
459:
460:     /* now at start good data, output file */
461:     /* until the ".end" is found (or EOF) */
462:     if ((fd = creat(filename,0644)) == ERROR)
463:     {
464:         printf("->ingresdi Can't Create %s\007\n",filename);
465:         printf("Assuming out of directory or space\n");
466:         exit();
467:     }
468:     for(ptr2 = buffer2, n2 = BYTES_PER_BLOCK; n2--;)
469:         *ptr2++ = '\0';
470:     ptr2 = buffer2;
471:     bytes_written = 0;
472:     rec_bytes = 0;
473:
474:     while((c=getabyte()) != EOF)
475:     {
476:         if ((c == '\r' || c == '\n' || c == '.') && lookatend())
477:         {
478:             /* flush <cr,lf, "."> */
479:             while((c=getabyte()) &&
480:                 c != EOF &&
481:                 (c == '\r' ||
482:                 c == '\n' ||
483:                 c == '.')) ;
484:             /* flush <"end",cr,lf> line */
485:             while((c=getabyte()) &&
486:                 c != EOF &&
487:                 c != '\r' &&
488:                 c != '\n') ;
489:             ungetbyte(c);
490:             break;
491:         }
492:
493:         else
494:         {
495:             *ptr2++ = c;
496:             bytes_written++;
497:             if (bytes_written == BYTES_PER_BLOCK)
498:                 writebuf2(fd);
499:             if (datasw)
500:             {
```

```
501:         rec_bytes++;
502:         if (rec_size == rec_bytes)
503:         {
504:             rec_bytes = 0;
505:             if ((c=getabyte()) && !(c == '\n' || c == '\r'))
506:                 ungetbyte(c);
507:             else if ((c=getabyte()) && !(c == '\n' || c == '\r'))
508:                 ungetbyte(c);
509:             else if ((c=getabyte()) && !(c == '\n' || c == '\r'))
510:                 ungetbyte(c);
511:             else if ((c=getabyte()) && !(c == '\n' || c == '\r'))
512:                 ungetbyte(c);
513:         }
514:     }
515: }
516: }
517: if (bytes_written) writebuf2(fd);
518: if (close(fd) == ERROR)
519: {
520:     printf("->ingresdi Can't Close %s\n",filename);
521:     exit();
522: }
523: }
524:
525: /*
526:  *****
527:  * generate ingres creat & copy specifications from file_def
528:  *****
529:  */
530: gen_ing()
531: {
532:
533:     strcpy(file_ing,wdn);
534:     strcat(file_ing,rel);
535:     strcat(file_ing,".ing");
536:     printf("->ingresdi generating %s\n",file_ing);
537:     if ((stream = fopen(file_ing,"w+")) == NULL)
538:     {
539:         printf("\007Unable to generate %s\n",file_ing);
540:         return(-1);
541:     }
542:     fprintf(stream,"destroy %s \\\g\n",rel);
543:     fprintf(stream,"create %s (\n",rel);
544:     for (current = 1; current <= last; current++)
545:     {
546:         attrcur = &attr[current];
547:         fprintf(stream,
548:             "    %s=c%d",
549:             attrcur->name,
550:             attrcur->frml);
```

```

551:         if (current == last)
552:             fprintf(stream, "\\)\n");
553:         else     fprintf(stream, ",\n");
554:     }
555:     fprintf(stream, "\\g\n");
556:     fprintf(stream, "copy  %s (\n", rel);
557:     for (current = 1; current <= last; current++)
558:     {
559:         attrcur = &attr[current];
560:         fprintf(stream,
561:             "        %s=c%d",
562:             attrcur->name,
563:             attrcur->frml);
564:         if (current == last)
565:             fprintf(stream, "\\) from \"%s\"\n", file_dat);
566:         else     fprintf(stream, ",\n");
567:     }
568:     fprintf(stream, "\\g\n");
569:     fclose(stream);
570:     return(0);
571: }
572:
573: /*
574:  *****
575:  *      generate mapping input instructions
576:  *****
577:  */
578: gen_mi()
579: {
580:     char file_mi[120];
581:
582:     s_mapper++;
583:     strcpy(file_mi, wdn);
584:     strcat(file_mi, rel);
585:     strcat(file_mi, ".mi");
586:     printf("->ingresdi generating %s\n", file_mi);
587:     if ((stream = fopen(file_mi, "w+")) == NULL)
588:     {
589:         printf("\007Unable to generate %s\n", file_mi);
590:         return(-1);
591:     }
592:     fprintf(stream, "destroy %s \\g\n", relation);
593:     fprintf(stream, "range of %s is %s \n", rel, rel);
594:     fprintf(stream, "retrieve into %s \\(\n", relation);
595:     for (current = 1; current <= last; current++)
596:     {
597:         attrcur = &attr[current];
598:         if (attrcur->frmt == 'c')
599:         {
600:             fprintf(stream, "        %s=%s.%s",

```

```

601:             attrcur->name,
602:             rel,
603:             attrcur->name);
604:         }
605:     else
606:     {
607:         if (attrcur->decimal == 0)
608:         {
609:             if (attrcur->frml <= 6)
610:                 fprintf(stream, "          %s=int2\\(%s.%s\\)",
611:                     attrcur->name,
612:                     rel,
613:                     attrcur->name);
614:             else
615:                 fprintf(stream, "          %s=int4\\(%s.%s\\)",
616:                     attrcur->name,
617:                     rel,
618:                     attrcur->name);
619:         }
620:     else
621:     {
622:         fprintf(stream, "          %s=float4\\(%s.%s\\)",
623:             attrcur->name,
624:             rel,
625:             attrcur->name);
626:     }
627: }
628:     if (current == last)
629:         fprintf(stream, "\\ \n");
630:     else
631:         fprintf(stream, ", \n");
632: }
633: fprintf(stream, "destroy %s \n", rel);
634: fprintf(stream, "\\g\n");
635: fclose(stream);
636: }
637: /*
638: *****
639: * collect attribute information into array "attr" from file_def
640: *****
641: */
642: collect()
643: {
644:     char line[100];
645:     char *ptr, *ptr2;
646:     int n, n2;
647:
648:     last = current = width = nonascii = 0;
649:     if ((filein = open(file_def, 0)) == ERROR)
650:     {

```

```

651:         printf("\n->ingresdi Can't Find %s\007\n",file_def);
652:         return(-1);
653:     }
654:     else
655:     {
656:         printf("->ingresdi processing %s\n",file_def);
657:         for(ptr=buffer, n2=BUFSIZE+10; n2--; )
658:             *ptr++ = '\0';
659:         n=read(filein,buffer+10,BUFSIZE);
660:         bytes_read = n ;
661:         if (!bytes_read)
662:         {
663:             for(ptr=buffer, n2=BUFSIZE+10; n2--; )
664:                 *ptr++ = '\0';
665:         }
666:         ptr = buffer+10;
667:         n = bytes_read;
668:     }
669:     while(*ptr)
670:     {
671:         /*get a line from file_def*/
672:         for(n2=0,ptr2=line,*ptr2 = '\0'; c = *ptr++ ; )
673:         {
674:             if (c == '\n') c = '\0';
675:             if (c != '\r')
676:             {
677:                 if (isupper(c))
678:                     *ptr2++ = tolower(c);
679:                 else
680:                     *ptr2++ = c;
681:             }
682:             *ptr2 = '\0';
683:             n-- ;
684:             n2++ ;
685:             if (n2 > 20)
686:             {
687:                 printf("\n->ingresdi CAN'T PROCESS %s\n",file_def);
688:                 exit(ERROR);
689:             }
690:             if (!n)
691:             {
692:                 for(ptr=buffer, n2=BUFSIZE+10; n2--; )
693:                     *ptr++ = '\0';
694:                 n=read(filein,buffer+10,BUFSIZE);
695:                 bytes_read = n ;
696:                 if (!bytes_read)
697:                 {
698:                     for(ptr=buffer, n2=BUFSIZE+10; n2--; )
699:                         *ptr++ = '\0';
700:                 }
701:                 ptr = buffer+10;

```

```

701:             n = bytes_read;
702:         }
703:         if (!c) break; /*end of line*/
704:     }
705:     if (!*line) continue; /*null line*/
706:
707:     /*get each field into the structure*/
708:     current++;
709:     last = current;
710:     attrcur = &attr[current];
711:     line[17] = '\0';
712:     ptr2=line+14; /* 3 digit decimal */
713:     attrcur->decimal = atoi(ptr2);
714:     line[14] = '\0';
715:     ptr2=line+11; /* 3 digit length */
716:     attrcur->frml = atoi(ptr2);
717:     attrcur->offset = width;
718:     width += attrcur->frml;
719:     ptr2=line+10; /* 1 character format */
720:     attrcur->frmt = *ptr2;
721:     if (!(*ptr2 == 'c' || *ptr2 == 'C')) nonascii++;
722:     *ptr2='\0'; /* stopper for end of name */
723:     if(*line)
724:     {
725:         while(line[(strlen(line)-1)] == ' ')
726:             line[(strlen(line)-1)] = '\0';
727:     }
728:     strcpy(attrcur->name,line);
729: } /*end of while not EOF */
730: close(filein);
731: }
732:
733: /*
734: *****
735: *      un_map      reforms data in floating point / integer fields.
736: *****
737: */
738: un_map()
739: {
740:     int i,ii,d,dd,dn,o;
741:     int n,nn;
742:     char fbuffer[BUFSIZ*8];
743:     char *ptr1, *prt2, *ptr3;
744:
745:     if ((fdat = fopen(file_dat,"r")) == NULL)
746:     {
747:         printf("\007->ingresdi Can't Read from %s\n",file_dat);
748:         exit(ERROR);
749:     }
750:     printf("->ingresdi processing %s [%d]\n",file_dat,width);

```

```

751:     strcpy(file_wrk,wdn);
752:     strcat(file_wrk,rel);
753:     strcat(file_wrk,".wrk");
754:     if ((fwrk = fopen(file_wrk,"w+")) == NULL)
755:     {
756:         printf("->ingresdi Can't write to %s\n\007",file_wrk);
757:         exit(ERROR);
758:     }
759:     printf("->ingresdi mapping into %s [%d]\n",file_wrk,width);
760:     for( ; (ii = fread(fbuffer, 1, width , fdat)) > 0; )
761:     {
762:         /*DEBUG printf("=>%s<=[%d]\n",fbuffer,ii);*/
763:         fbuffer[ii] = '\0';
764:         /* left adjust & un-zero fill int and float fields */
765:         for(current = 1; current <= last; current++)
766:         {
767:             attrcur = &attr[current];
768:             if (attrcur->fmt == 'n' ||
769:                 attrcur->fmt == 'N')
770:             {
771:                 i=attrcur->fml;
772:                 d=attrcur->decimal;
773:                 o=attrcur->offset;
774:                 n=0;           /* beg */
775:                 nn=n + (i-1); /* end */
776:
777:                 /****test for decimal point in data****/
778:                 for(dn = dd = 0 ; nn >= n ; nn-- )
779:                 {
780:                     if (fbuffer[nn] == '.')
781:                     {
782:                         nn = 0;
783:                         dn = 1;
784:                     }
785:                     else dd++ ;
786:                 }
787:
788:                 /****if decimal specified in data ****/
789:                 /****only left adjust the field ****/
790:                 if (dn)
791:                 {
792:                     i=attrcur->fml;
793:                     d=attrcur->decimal;
794:                     o=attrcur->offset;
795:                     n=0;           /* beg */
796:                     nn=n + (i-1); /* end */
797:                     while((nn >= n) && (fbuffer[n] == ' '))
798:                         n++ ;
799:                     while(nn >= n)
800:                     {

```

```
801:         fbuffer[o] = fbuffer[n] ;
802:         o++ ;
803:         n++ ;
804:     }
805:     while(nn >= o)
806:     {
807:         fbuffer[o] = ' ' ;
808:         o++ ;
809:     }
810:     } /*end decimal in data*/
811:
812:     /****if no decimal in data***/
813:     /****revise the data accordingly***/
814:     /****use implied decimal position***/
815:     if (!dn)
816:     {
817:         i=attrcur->frml;
818:         d=attrcur->decimal;
819:         o=attrcur->offset;
820:         n=o;          /* beg */
821:         nn=n + (i-1); /* end */
822:         if (fbuffer[n] == '-')
823:         {
824:             n++ ;
825:             o++ ;
826:         }
827:         while((nn >= n) &&
828:             ((fbuffer[n] == '0') ||
829:              (fbuffer[n] == ' ')))
830:         {
831:             if (d && (n == (nn-d)))
832:             {
833:                 fbuffer[o] = '.';
834:                 o++ ;
835:                 n++ ;
836:                 break;
837:             }
838:             else n++ ;
839:         }
840:         while(nn >= n)
841:         {
842:             fbuffer[o] = fbuffer[n] ;
843:             o++ ;
844:             if (d && (n == (nn-d)))
845:             {
846:                 fbuffer[o] = '.';
847:                 o++ ;
848:                 n++ ;
849:             }
850:             else n++ ;
```

```

851:                                     }
852:                                     while (nn >= 0)
853:                                     {
854:                                         fbuffer[o] = ' ';
855:                                         o++;
856:                                     }
857:                                     } /*end no decimal in data*/
858:
859:                                     } /*end of each numeric field*/
860:                                     } /* end of all fields */
861:                                     fwrite(fbuffer, 1, ii, fwrk) ;
862:                                     /*DEBUG printf("=>%s<={%d}\n\n",fbuffer,ii); */
863:                                     }
864:                                     fclose(fdat);
865:                                     fclose(fwrk);
866:                                     if (unlink(file_dat))
867:                                         printf("\n->ingresdi Can't unlink %s\007\n",file_dat);
868:                                     if (link(file_wrk,file_dat))
869:                                         printf("\n->ingresdi Can't rename %s\007\n",file_wrk);
870:                                     }
871:
872: /*
873: *****
874: *      lookatend      looks for '.end' statement
875: *****
876: */
877: int
878: lookatend()
879: {
880:     char work[20];
881:     char *wk;
882:     int w;
883:     for (wk = work, w = 11; w-- ; )
884:         *wk++ = '\0' ;
885:     for (wk = work, w = 10; w-- ; )
886:         *wk++ = getabyte() ;
887:     for (w = 10; w-- ; )
888:     {
889:         wk-- ;
890:         ungetbyte(*wk) ;
891:     }
892:     wk = work;
893:     while ((*wk == '\r') ||
894:            (*wk == '\n') ||
895:            (*wk == '.'))
896:         wk++;
897:     if (*wk++ == 'e' &&
898:        *wk++ == 'n' &&
899:        *wk++ == 'd' &&
900:        (*wk == '\r' || *wk == '\n'))

```

```
901:         return(1);
902:     else    return(0);
903: }
904:
905: /*
906:  *****
907:  *      getabyte    returns next byte from file buffer
908:  *****
909:  */
910: char
911: getabyte()
912: {
913:     char c;
914:     if (!bytes_read)
915:     {
916:         n=read(filein,buffer+10,BUFSIZE);
917:         bytes_read = n;
918:         ptr = buffer+10;
919:     }
920:     if (!bytes_read)
921:     {
922:         signal_eof++;
923:         return(EOF);
924:     }
925:     bytes_read--;
926:     c = *ptr++;
927:     return(c);
928: }
929: /*
930:  *****
931:  *      ungetbyte    puts a byte back into the file buffer
932:  *****
933:  */
934: char
935: ungetbyte(c)
936: char c;
937: {
938:     bytes_read++;
939:     ptr--;
940:     *ptr = c;
941:     return(c);
942: }
943:
944: /*
945:  *****
946:  *      writebuf2    writes output buffer2 and preps for next block
947:  *****
948:  */
949: int
950: writebuf2(fd)
```

```
951: int fd;
952: {
953:     int n2;
954:
955:     if (write(fd,buffer2,bytes_written) != bytes_written)
956:     {
957:         printf("\n\007->ingresdi Write Error\n");
958:         exit(ERROR);
959:     }
960:     for(ptr2 = buffer2, n2 = BYTES_PER_BLOCK; n2--; )
961:         *ptr2++ = '\0';
962:     ptr2 = buffer2;
963:     bytes_written = 0;
964: }
965: /*
966: *****
967: * END OF SOURCE
968: *****
969: */
```

Characters: 32282

SECTION 2.7

File: INGRESUL.C

Size: 11K

Page: 1

```

1:
2: *****
3: * ingresul.c will combine files into a single stream for
4: *      upload to INGRES on the VAX
5: *****
6: *
7: *  PROGRAMS RUNS ON CP/M MICROCOMPUTER
8: *
9: *      The input files used are:
10: *
11: *          1: <relation>.ING ingres create/copy command
12: *          2: <relation>.MI  ingres mapper input command
13: *          3: <relation>.MO  ingres mapper output command
14: *          4: <relation>.DEF dBASEIII create definitions
15: *          5: <relation>.DAT actual data file to be used
16: *
17: *      The output file is:
18: *
19: *          -> <relation>.S   the output file created
20: *
21: *      If everything is ok, the input files will be deleted.
22: *
23: *
24: * NOTE:  THIS CODE WAS WRITTEN AND TESTED FOR BDS-C v1.50
25: *        THE OUTPUT FILES ARE FORMATTED FOR INPUT TO
26: *        UNIX VERSION OF RTI INGRES  v1.2
27: *
28: * SAMPLE OUTPUT STREAM:
29:
30: .ingres clist 30
31: create clist (
32:     code=c5,
33:     name=c15,
34:     bldg=c5,
35:     room=c5)
36: copy clist (
37:     code=c5,
38:     name=c15,
39:     bldg=c5,
40:     room=c5) from "clist.dat"
41: .end
42: .dbase clist 30
43: code      c 5 0
44: name      c 15 0
45: bldg      c 5 0
46: room      c 5 0
47: .end
48: .data clist 30
49: 8241 Miyashiro      33  0047
50: 8241 Woods, Bev     33  0067

```

```
51: 8254 Campbell      33  0055
52: 8241 Landa        33  0065
53: .end
54: *
55: *****
56: */
57:
58: #include <bdscio.h>
59: #include <hardware.h>
60:
61: #define ERROR -1
62: #define BUFSIZE 512
63: #define BYTES_PER_BLOCK 128
64: #define BLOCKS 4
65: #define MAXATTR 32
66: #define MAXNAME 10
67:
68: struct attrib
69: {
70:     char name[MAXNAME+1];
71:     char frmt;
72:     int frml;
73:     int decimal;
74:     int offset;
75: } attr[MAXATTR], *attrcur;
76:
77: int current, last, width, nonascii;
78: int s_dbase, s_data, s_ingres, s_mapper;
79:
80: int filein, fileout;
81: int n, n2;
82: int bytes_read, bytes_written;
83: int rec_size, rec_bytes;
84: int signal_eof;
85:
86: char work[30];
87: char file_s[20];
88: char file_ing[20];
89: char file_def[20];
90: char file_dat[20];
91: char file_cmd[20];
92: char file_mo[20];
93: char file_mi[20];
94: char buffer[BUFSIZE];
95: char buffer2[BYTES_PER_BLOCK];
96:
97: char c;
98: char *ptr;          /* input stream buffer pointer */
99: char *ptr2;         /* output file buffer pointer */
100: char *database;     /* pointer to the database name */
```

```

101: char *relation;          /* pointer to the relation name */
102: char *indexer;           /* pointer to the index field name */
103:
104: char getabyte();
105:
106: /*
107:  *****
108:  * MAIN CONTROL SECTION
109:  *****
110:  */
111: main (argc,argv)
112: int  argc;
113: char *argv[];
114: {
115:     if (argc != 3)
116:     {
117:         printf("Usage: ingresul <database> <relation> \n");
118:         printf("where      <database> name of an existing database\n");
119:         printf("      <relation> name of a relation in database\n");
120:         exit(-1);
121:     }
122:
123:     database = argv[1];
124:     relation = argv[2];
125:     indexer = "NAME";
126:
127:     printf("->ingresul starting\n");
128:     setup();
129:     collect();
130:     printf("->ingresul building %s\n",file_s);
131:     opener();
132:     if (do_thedata(file_def,1) != ERROR) s_dbase++;
133:     if (do_thedata(file_ing,2) != ERROR) s_ingres++;
134:     if (do_thedata(file_mi,4) != ERROR) s_mapper++;
135:     if (do_thedata(file_mo,4) != ERROR) s_mapper++;
136:     if (do_thedata(file_dat,3) != ERROR) s_data++;
137:     if (bytes_written) writebuf2(fileout);
138:     if (close(fileout) == ERROR)
139:     {
140:         printf("->ingresul Can't Close %s\n",file_s);
141:         exit(ERROR);
142:     }
143:     else
144:     {
145:         if (!s_dbase || !s_data)
146:         {
147:             printf("\007Not enough information to complete %s\n",
148:                 file_s);
149:             unlink(file_s);
150:             exit(ERROR);

```

```

151:         }
152:     else
153:     {
154:         if (s_dbase) unlink(file_def);
155:         if (s_data) unlink(file_dat);
156:         if (s_ingres) unlink(file_ing);
157:         if (s_mapper) unlink(file_mi);
158:         if (s_mapper) unlink(file_mo);
159:         printf("->ingresul ok\n");
160:     }
161: }
162: }
163:
164: /*
165:  *****
166:  *      do_thedata  writes current file to output buffer
167:  *****
168:  */
169: int
170: do_thedata(filename,datasw)
171: char *filename;
172: int datasw;
173: {
174:     int fd, n, n2;
175:     char *hdr;
176:
177:     switch(datasw)
178:     {
179:         case 1 : hdr = ".dbase"; break;
180:         case 2 : hdr = ".ingres"; break;
181:         case 3 : hdr = ".data"; break;
182:         case 4 : hdr = ".mapper"; break;
183:         default: hdr = "";
184:     }
185:     if ((filein = open(filename,0)) == ERROR)
186:     {
187:         printf("->ingresul not using %s\n",filename);
188:         return(ERROR);
189:     }
190:     else
191:     {
192:         printf("->ingresul reading %s\n",filename);
193:         puthdr(hdr,filename,datasw);
194:         n=read(filein,buffer,BLOCKS);
195:         bytes_read = (n * BYTES_PER_BLOCK);
196:         ptr = buffer;
197:     }
198:     /* correction for leading <cr,lf> if file originated
199:        with dBASE II. */
200:     if ((datasw == 1 || datasw == 3) &&

```

```
201:      ((*ptr = '\r') && (*(ptr+1) = '\n'))
202:      {
203:          ptr = ptr+2;
204:          bytes_read = bytes_read - 2;
205:      }
206:      while((c=getabyte()) && c != CPMEOF)
207:      {
208:          *ptr2++ = c;
209:          bytes_written++;
210:          if (bytes_written == BYTES_PER_BLOCK)
211:              writebuf2(fileout);
212:      }
213:      putend();
214: }
215:
216: /*
217: *****
218: *      setup      setup file names, open and prime output stream
219: *****
220: */
221: int
222: setup()
223: {
224:     strcpy(file_s,relation);
225:     strcat(file_s, ".S");
226:     strcpy(file_ing,relation);
227:     strcat(file_ing, ".ING");
228:     strcpy(file_def,relation);
229:     strcat(file_def, ".DEF");
230:     strcpy(file_dat,relation);
231:     strcat(file_dat, ".DAT");
232:     strcpy(file_mi, relation);
233:     strcat(file_mi, ".MI ");
234:     strcpy(file_mo, relation);
235:     strcat(file_mo, ".MO ");
236: }
237:
238: /*
239: *****
240: *      opener      open and prime output stream
241: *****
242: */
243: int
244: opener()
245: {
246:     if ((fileout = creat(file_s)) == ERROR)
247:     {
248:         printf("\nCan't Create %s\007\n",file_s);
249:         exit(ERROR);
250:     }
```

```

251:     else
252:     {
253:         for (ptr2 = buffer2, n2 = BYTES_PER_BLOCK; n2--; )
254:             *ptr2++ = CPMEOF;
255:         ptr2 = buffer2;
256:         bytes_written = 0;
257:     }
258: }
259:
260: /*
261:  *****
262:  *      puthdr      writes a correct header to the output file stream
263:  *****
264:  */
265: int
266: puthdr(hdr, filename, datasw)
267: char *hdr;
268: char *filename;
269: int  datasw;
270: {
271:     if (nonascii)
272:     {
273:         if (datasw == 4)
274:             sprintf(buffer, "%s r %s r %s \r\n", hdr, relation, filename);
275:         else sprintf(buffer, "%s r %s %d \r\n", hdr, relation, width);
276:     }
277:     else
278:     {
279:         if (datasw == 4)
280:             sprintf(buffer, "%s %s %s \r\n", hdr, relation, filename);
281:         else sprintf(buffer, "%s %s %d \r\n", hdr, relation, width);
282:     }
283:     bytes_read = strlen(buffer);
284:     ptr = buffer;
285:     while (bytes_read--)
286:     {
287:         if (isupper(*ptr))
288:             *ptr2++ = tolower(*ptr++);
289:         else *ptr2++ = *ptr++;
290:         bytes_written++;
291:         if (bytes_written == BYTES_PER_BLOCK)
292:             writebuf2(fileout);
293:     }
294: }
295:
296: /*
297:  *****
298:  *      putend      writes a correct .end to the output file stream
299:  *****
300:  */

```

```

301: int
302: putend()
303: {
304:     sprintf(buffer, ".end\r\n");
305:     bytes_read = strlen(buffer);
306:     ptr = buffer;
307:     while(bytes_read--)
308:     {
309:         *ptr2++ = *ptr++;
310:         bytes_written++;
311:         if (bytes_written == BYTES_PER_BLOCK)
312:             writebuf2(fileout);
313:     }
314: }
315:
316: /*
317: *****
318: * collect attribute information into array "attr"
319: *****
320: */
321: int
322: collect()
323: {
324:     char line[100];
325:     char *ptr, *ptr2;
326:     int n, n2;
327:
328:     last = current = width = nonascii = 0;
329:     if ((filein = open(file_def, 0)) == ERROR)
330:     {
331:         printf("\n->ingresul Can't Find %s\007\n", file_def);
332:         exit(ERROR);
333:     }
334:     else
335:     {
336:         printf("->ingresul processing %s\n", file_def);
337:         n = read(filein, buffer, BLOCKS);
338:         bytes_read = (n * BYTES_PER_BLOCK);
339:         if (!bytes_read)
340:         {
341:             for(ptr = buffer, n2 = BYTES_PER_BLOCK; n2--; )
342:                 *ptr++ = CPMEOF;
343:         }
344:         ptr = buffer;
345:         n = bytes_read;
346:     }
347:     while(*ptr != CPMEOF)
348:     {
349:         /*get a line from file def (character cannot be null or eof) */
350:         for(ptr2 = line, n2 = 100; n2--; *ptr2++ = '\0') ;

```

```

351:     for(n2=0,ptr2=line ; c = *ptr++ , c != CPMEOF ; )
352:     {
353:         if (c == '\n') c = '\0';
354:         if (c != '\r') *ptr2++ = c;
355:         n-- ;
356:         n2++ ;
357:         if (n2 > 20)
358:         {
359:             printf("\7->ingresul CAN'T PROCESS %s\n",file_def);
360:             exit(ERROR);
361:         }
362:         if (!n)
363:         {
364:             n=read(filein,buffer,BLOCKS);
365:             bytes_read = (n * BYTES_PER_BLOCK);
366:             if (!bytes_read)
367:             {
368:                 for(ptr=buffer, n2=BYTES_PER_BLOCK; n2--; )
369:                     *ptr++ = CPMEOF;
370:             }
371:             ptr = buffer;
372:             n = bytes_read;
373:         }
374:         if (!c) break; /*end of line*/
375:     }
376:     /*DEBUG printf(">>%s<< %d %d\n",line,n,n2); */
377:     if (!*line) continue; /*null line*/
378:
379:     /*get each field into the structure*/
380:     current++;
381:     last = current;
382:     attrcur = &attr[current];
383:     line[17] = '\0';
384:     ptr2=line+14; /* 3 digit decimal */
385:     attrcur->decimal = atoi(ptr2);
386:     line[14] = '\0';
387:     ptr2=line+11; /* 3 digit length */
388:     attrcur->frml = atoi(ptr2);
389:     attrcur->offset = width;
390:     width += attrcur->frml;
391:     ptr2=line+10; /* 1 character format */
392:     attrcur->frmt = *ptr2;
393:     if (!(*ptr2 == 'c' || *ptr2 == 'C')) nonascii++;
394:     *ptr2='\0'; /* stopper for end of name */
395:     if(*line)
396:     {
397:         while(line[(strlen(line)-1)] == ' ')
398:             line[(strlen(line)-1)] = '\0';
399:     }
400:     strcpy(attrcur->name,line);

```

```
401:         } /*end of while not CPMEOF */
402:         close(filein);
403:     }
404:
405: /*
406: *****
407: *      getabyte      returns next byte from file buffer
408: *****
409: */
410: char
411: getabyte()
412: {
413:     char c;
414:     if (!bytes_read)
415:     {
416:         n=read(filein,buffer,BLOCKS);
417:         bytes_read = (n * BYTES_PER_BLOCK);
418:         ptr = buffer;
419:     }
420:     if (!bytes_read)
421:     {
422:         signal_eof++;
423:         return(CPMEOF);
424:     }
425:     bytes_read--;
426:     c = *ptr++;
427:     return(c);
428: }
429:
430: /*
431: *****
432: *      writebuf2     writes output buffer2 and preps for next block
433: *****
434: */
435: int
436: writebuf2(fd)
437: int fd;
438: {
439:     int n2;
440:
441:     if (write(fd,buffer2,1) != 1)
442:     {
443:         printf("\n->ingresul Write Error: %d\n",tell(fd));
444:         puts(errmsg(errno()));
445:         exit();
446:     }
447:     for(ptr2 = buffer2, n2 = BYTES_PER_BLOCK; n2--; )
448:         *ptr2++ = CPMEOF;
449:     ptr2 = buffer2;
450:     bytes_written = 0;
```

```
451: }  
452: /*  
453: *****  
454: * END OF SOURCE  
455: *****  
456: */
```

Characters: 10765

```

1:
2: *****
3: * ingresdb.c will break out the job stream created by
4: *   getingdat on the VAX into 4 files:
5: *****
6: *
7: * PROGRAM RUNS ON CP/M MICROCOMPUTER
8: *
9: *   The input file is <relation>.S stream downloaded from
10: *       INGRES on VAX
11: *
12: *   The files produced are:
13: *       1: <relation>.ING ingres create/copy command
14: *       2: <relation>.MI ingres re-mapping instructions
15: *       3: <relation>.DEF dBASEII create definitions
16: *       4: <relation>.DAT actual data file to be used
17: *       5: INGRES.CMD GLOBAL module to bootstrap dbase
18: *
19: * NOTE: THIS CODE WAS WRITTEN AND TESTED FOR BDS-C v1.50
20: *       THE OUTPUT FILES ARE FORMATTED FOR INPUT TO
21: *       dBASEII v2.3B
22: *
23: * SAMPLE DATA FILE INPUT STREAM
24: *
25: *
26:
27: .ingres clist 30
28: create clist (
29:     code=c5,
30:     name=cl5,
31:     bldg=c5,
32:     room=c5)
33: copy clist (
34:     code=c5,
35:     name=cl5,
36:     bldg=c5,
37:     room=c5) from "clist.dat"
38: .end
39: .dbase clist 30
40: code      c  5  0
41: name      c 15  0
42: bldg      c  5  0
43: room      c  5  0
44: .end
45: .data clist 30
46: 8241 Miyashiro      33  0047
47: 8241 Woods, Bev     33  0067
48: 8254 Campbell       33  0055
49: 8241 Landa           33  0065
50: .end

```

```

51:  *
52:  *****
53:  */
54:  #include <bdscio.h>
55:  #include <hardware.h>
56:
57:  #define BUFSIZE 512
#define BYTES_PER_BLOCK 128
58:  #define BLOCKS 4
59:  #define MAXNAME 10
60:  #define MAXATTR 32
61:
62:  struct attrib
63:  {
64:      char name[MAXNAME+1];
65:      char frmt;
66:      int frml;
67:  } attr[MAXATTR], *attrcur;
68:
69:  int current, last, width, nonascii;
70:  int s_dbase, s_data, s_ingres, s_mapper;
71:
72:  int filein, fileout;
73:  int n, n2;
74:  int bytes_read, bytes_written;
75:  int rec_size, rec_bytes;
76:  int signal_eof;
77:
78:  char work[30];
79:  char rel[20];
80:  char file_s[20];
81:  char file_ing[20];
82:  char file_def[20];
83:  char file_dat[20];
84:  char file_map[20];
85:  char file_cmd[20];
86:  char buffer[BUFSIZE+10];
87:  char buffer2[BYTES_PER_BLOCK];
88:
89:  char c;
90:  char *ptr;          /* input stream buffer pointer */
91:  char *ptr2;         /* output file buffer pointer */
92:  char *database;     /* pointer to the database name */
93:  char *relation;     /* pointer to the relation name */
94:  char *datafile;     /* pointer to the datafile name */
95:  char *indexer;      /* pointer to the index field name */
96:
97:  char getabyte();
98:  /*
99:  *****
100:  *      main control section is a loop which scans the input stream

```

```

101: *      for ".dbase", ".ingres", or ".data" section headings.
102: *      For each section an appropriate call is made to
103: *      "do_thedata()" to extract the data for this section.
104: *      At the end of the input stream "do_global()" is called.
105: *****
106: */
107: main (argc,argv)
int argc;
108: char *argv[];
109: {
110:
111:     if (argc > 4 || argc < 3)
112:     {
113:         printf("Usage: ingresdb <database> <relation> [index-name]\n");
114:         printf("where          <database> name of an existing database\n");
115:         printf("          <relation> name of a relation in database\n");
116:         printf("          [index-name] field to be used for indexing\n");
117:         exit(-1);
118:     }
119:     database = argv[1];
120:     relation = argv[2];
121:     strcpy(rel,argv[2]);
122:     if (argc == 4)
123:         indexer = argv[3];
124:     else     indexer = " ";
125:     setup();
126:     while((c=getabyte()) != CPMEOF)
127:     {
128:         if (c == '.')
129:         {
130:             ptr2 = work;
131:             *ptr2 = '\0';
132:             n2 = 25;
133:             while(n2-- && (c=getabyte()) &&
134:                 c != CPMEOF &&
135:                 c != '\n' &&
136:                 c != ' ')
137:             {
138:                 *ptr2++ = c;
139:                 *ptr2 = '\0';
140:             }
141:             if (!strcmp(work,"mapper"))
142:             {
143:                 s_mapper++;
144:                 do_mapper();
145:             }
146:
147:             if (!strcmp(work,"dbase"))
148:             {
149:                 s_dbase++;
150:                 do_dbase();

```

```
151:         }
152:
153:         if (!strcmp(work,"ingres"))
154:         {
155:             s_ingres++;
156:             do_ingres();
157:         }
158:
159:         if (!strcmp(work,"data"))
160:         {
161:             s_data++;
162:             do_data();
163:         } /*end of *ptr = '.' */
164:     } /*end while getabyte */
165:
166:     if (!s_dbase)
167:     {
168:         printf("\007Can't boot dbase because %s ",file_s);
169:         printf("has no data for %s\n",file_def);
170:     }
171:     if (!s_data)
172:     {
173:         printf("\007Can't boot dbase because %s ",file_s);
174:         printf("has no data for %s\n",file_dat);
175:     }
176:     if (!s_dbase || !s_data)
177:     {
178:         printf("\007Removing Old dbase bootstrap %s\n",file_cmd);
179:         unlink(file_cmd);
180:         exit(-1);
181:     }
182:     printf("->ingresdb building file INGRES.CMD\n");
183:     do_global();
184:     printf("->ingresdb ok\n");
185: }
186: /*
187: *****
188: *      setup      setup file names, open and prime input stream
189: *****
190: */
191: setup()
192: {
193:     strcpy(file_s,rel);
194:     strcat(file_s,".S");
195:     strcpy(file_def,rel);
196:     strcat(file_def,".DEF");
197:     strcpy(file_dat,rel);
198:     strcat(file_dat,".DAT");
199:     strcpy(file_cmd,"INGRES.CMD");
200:
```

```

201:         if ((filein = open(file_s,0)) == ERROR)
202:         {
203:             printf("\nCan't Find %s\007\n",file_s);
204:             exit(-1);
205:         }
206:         else
207:         {
208:             n=read(filein,buffer+10,BLOCKS);
209:             bytes_read = (n * BYTES_PER_BLOCK);
210:             ptr = buffer+10;
211:         }
212: /*
213: *****
214: *      do_global      writes the global boot strap file "INGRES.CMD"
215: *****
216: */
217: int
218: do_global()
219: {
220:     struct _buf iobuf[BUFSIZ];
221:
222:     strcpy(file_cmd,"INGRES.CMD");
223:     if (fcreat(file_cmd,iobuf) == ERROR)
224:     {
225:         printf("\007\nCan't Create %s\n",file_cmd);
226:         printf("Assuming out of directory or space\n");
227:         exit(-1);
228:     }
229:     fprintf(iobuf,"* *****\n");
230:     fprintf(iobuf,"* BOOTSTRAP FOR: INGRES %s %s\n",database,rel);
231:     fprintf(iobuf,"* this file is created dynamically by ingresdb.c\n");
232:     fprintf(iobuf,"* *****\n");
233:     fprintf(iobuf,"SET TALK OFF\n");
234:     fprintf(iobuf,"ERASE\n");
235:     fprintf(iobuf,"@ 10,14 SAY \"DBASE INGRES %s %s\"\\n",
236:             database,rel);
237:     fprintf(iobuf,"STORE \"%s\" TO DATABASE\\n",database);
238:     fprintf(iobuf,"STORE \"%s\" TO RELATION\\n",rel);
239:     fprintf(iobuf,"STORE \"%s\" TO INDEXER\\n",indexer);
240:     fprintf(iobuf,"DO INGRESMM\\n");
241:     fprintf(iobuf,"RETURN\\n");
242:     for(n=0;n<25;n++) work[n] = CPMEOF;
243:     work[n] = '\0';
244:     fprintf(iobuf,"%s",work);
245:     fclose(iobuf);
246: }
247: /*
248: *****
249: *      do_mapper      set up for mapper files from input stream
250: *****

```

```
251:  */
252: do_mapper()
253: {
254:     char fil[60], *ex;
255:     char c;
256:
257:     /* flush white space */
258:     while((c=getabyte()) && c != CPMEOF && c == ' ');
259:     ungetbyte(c);
260:     if (c == CPMEOF) return;
261:     /* extract relation name */
262:     ex = fil;
263:     while((c=getabyte()) && c != CPMEOF && c != '\n' && c != ' ')
264:     {
265:         if (islower(c))
266:             *ex++ = toupper(c);
267:         else
268:             *ex++ = c;
269:     }
270:     *ex = '\0';
271:     if (fil[0] == 'R' && fil[1] == '_')
272:         strcpy(rel,&fil[2]);
273:     else
274:         strcpy(rel,fil);
275:     if (c == CPMEOF) return;
276:     /* flush white space */
277:     while((c=getabyte()) && c != CPMEOF && c == ' ');
278:     ungetbyte(c);
279:     if (c == CPMEOF) return;
280:     /* extract file name */
281:     ex = fil;
282:     while((c=getabyte()) && c != CPMEOF && c != '\n' && c != ' ')
283:     {
284:         if (islower(c))
285:             *ex++ = toupper(c);
286:         else
287:             *ex++ = c;
288:     }
289:     *ex = '\0';
290:     if (c == CPMEOF) return;
291:     /* flush single occurrence of <space><nl> */
292:     if ((c=getabyte()) && c != ' ')
293:         ungetbyte(c);
294:     if ((c=getabyte()) && !(c == '\n' || c == '\r'))
295:         ungetbyte(c);
296:     if ((c=getabyte()) && !(c == '\n' || c == '\r'))
297:         ungetbyte(c);
298:     if (c == CPMEOF) return;
299:     /* create & copy data to the requested file */
300:     if (fil[0] == 'R' && fil[1] == '_')
301:         strcpy(file_map,&fil[2]);
302:     else
303:         strcpy(file_map,fil);
304:     printf("->ingresdb building %s\n",file_map);
305:     do_thedata(file_map,0);
```

```

301: }
302: /*
303: *****
304: *      do ingres      set up for ingres files from input stream
305: *****
306: */
307: do_ingres()
{
308:     char fil[60], *ex;
309:     char c;
310:
311:     /* flush white space */
312:     while((c=getabyte()) && c != CPMEOF && c == ' ');
313:     ungetbyte(c);
314:     if (c == CPMEOF) return;
315:     /* extract relation name */
316:     ex = fil;
317:     while((c=getabyte()) && c != CPMEOF && c != '\n' && c != ' ')
318:     {
319:         if (islower(c))
320:             *ex++ = toupper(c);
321:         else *ex++ = c;
322:     }
323:     *ex = '\0';
324:     if (fil[0] == 'R' && fil[1] == '_')
325:         strcpy(rel,&fil[2]);
326:     else strcpy(rel,fil);
327:     if (c == CPMEOF) return;
328:     /* flush white space */
329:     while((c=getabyte()) && c != CPMEOF && c == ' ');
330:     ungetbyte(c);
331:     if (c == CPMEOF) return;
332:     /* extract file name */
333:     ex = fil;
334:     while((c=getabyte()) && c != CPMEOF && c != '\n' && c != ' ')
335:     {
336:         if (islower(c))
337:             *ex++ = toupper(c);
338:         else *ex++ = c;
339:     }
340:     *ex = '\0';
341:     if (c == CPMEOF) return;
342:     rec size = atoi(fil);
343:     /* Flush single occurrence of <space><nl> */
344:     if ((c=getabyte()) && c != ' ')
345:         ungetbyte(c);
346:     if ((c=getabyte()) && !(c == '\n' || c == '\r'))
347:         ungetbyte(c);
348:     if ((c=getabyte()) && !(c == '\n' || c == '\r'))
349:         ungetbyte(c);
350:     if (c == CPMEOF) return;

```

```

351:      /* create & copy data to the requested file*/
352:      strcpy(file_ing,rel);
353:      strcat(file_ing,".ING");
354:      printf("->ingresdb building %s\n",file_ing);
355:      do_thedata(file_ing,0);
356:  }
357:  /*
*****
358:  *      do dbase      set up for dbase files from input stream
359:  *****
360:  */
361:  do_dbase()
362:  {
363:      char fil[60], *ex;
364:      char c;
365:
366:      /* flush white space */
367:      while((c=getabyte()) && c != CPMEOF && c == ' ');
368:      ungetbyte(c);
369:      if (c == CPMEOF) return;
370:      /* extract relation name */
371:      ex = fil;
372:      while((c=getabyte()) && c != CPMEOF && c != '\n' && c != ' ')
373:      {
374:          if (islower(c))
375:              *ex++ = toupper(c);
376:          else *ex++ = c;
377:      }
378:      *ex = '\0';
379:      if (fil[0] == 'R' && fil[1] == '_')
380:          strcpy(rel,&fil[2]);
381:      else strcpy(rel,fil);
382:      if (c == CPMEOF) return;
383:      /* flush white space */
384:      while((c=getabyte()) && c != CPMEOF && c == ' ');
385:      ungetbyte(c);
386:      if (c == CPMEOF) return;
387:      /* extract data relation size */
388:      ex = fil;
389:      while((c=getabyte()) && c != CPMEOF && c != '\n' && c != ' ')
390:      {
391:          if (islower(c))
392:              *ex++ = toupper(c);
393:          else *ex++ = c;
394:      }
395:      *ex = '\0';
396:      if (c == CPMEOF) return;
397:      rec_size = atoi(fil);
398:      /* Flush single occurrence of <space><nl> */
399:      if ((c=getabyte()) && c != ' ')
400:          ungetbyte(c);

```

```

401:         if ((c=getabyte()) && !(c == '\n' || c == '\r'))
402:             ungetbyte(c);
403:         if ((c=getabyte()) && !(c == '\n' || c == '\r'))
404:             ungetbyte(c);
405:         if (c == CPMEOF) return;
406:         /* create & copy data to the requested file*/
407:         strcpy(file_def,rel);
         strcat(file_def,".DEF");
408:         printf("->ingresdb building %s\n",file_def);
409:         do_thedata(file_def,0);
410:     }
411: /*
412: *****
413: *      do data      set up for data files from input stream
414: *****
415: */
416: do_data()
417: {
418:     char fil[60], *ex;
419:     char c;
420:
421:     /* flush white space */
422:     while((c=getabyte()) && c != CPMEOF && c == ' ');
423:     ungetbyte(c);
424:     if (c == CPMEOF) return;
425:     /* extract relation name */
426:     ex = fil;
427:     while((c=getabyte()) && c != CPMEOF && c != '\n' && c != ' ')
428:     {
429:         if (islower(c))
430:             *ex++ = toupper(c);
431:         else
432:             *ex++ = c;
433:     }
434:     *ex = '\0';
435:     if (fil[0] == 'R' && fil[1] == '_')
436:         strcpy(rel,fil);
437:     else
438:         strcpy(rel,fil);
439:     if (c == CPMEOF) return;
440:     /* flush white space */
441:     while((c=getabyte()) && c != CPMEOF && c == ' ');
442:     ungetbyte(c);
443:     if (c == CPMEOF) return;
444:     /* extract data relation size */
445:     ex = fil;
446:     while((c=getabyte()) && c != CPMEOF && c != '\n' && c != ' ')
447:     {
448:         if (islower(c))
449:             *ex++ = toupper(c);
450:         else
451:             *ex++ = c;
452:     }
453:     *ex = '\0';

```

```

451:         if (c == CPMEOF) return;
452:         rec_size = atoi(fil);
453:         /* flush single occurrence of <space><nl> */
454:         if ((c=getabyte()) && c != ' ')
455:             ungetbyte(c);
456:         if ((c=getabyte()) && !(c == '\n' || c == '\r'))
457:             ungetbyte(c);
458:         if ((c=getabyte()) && !(c == '\n' || c == '\r'))
459:             ungetbyte(c);
460:         if (c == CPMEOF) return;
461:         /* create & copy data to the requested file*/
462:         strcpy(file_dat,rel);
463:         strcat(file_dat,".DAT");
464:         printf("->ingresdb building %s [%d]\n",file_dat,rec_size);
465:         do_thedata(file_dat,1);
466:     }
467:     /*
468:     *      do_thedata   writes current input stream to output buffer
469:     *
470:     *      NOTE: this routine is based upon this format:
471:     *             see sample at front of this listing.
472:     *
473:     *             ".ingres<space>name<space>size-nnn<space><cr,lf>"
474:     *             "<a stream of actual data (of record size-nnn)>"
475:     *             "<cr,lf>.end<cr,lf>"
476:     *
477:     *      ALL DATA AFTER THE '.ingres' LINE AND UPTO THE '.end'
478:     *      WILL BE WRITTEN TO THE OUTPUT FILE NAMED.
479:     *
480:     *      If 'datasw' is true, a standard CP/M <cr,lf> will be added
481:     *      after every 'rec_size' bytes of data read in the ".data"
482:     *      section.
483:     *
484:     *****
485:     */
486:     int
487:     do_thedata(filename,datasw)
488:     char *filename;
489:     int datasw;
490:     {
491:         char c;
492:         int fd, n2;
493:
494:         /* now at start good data, output file */
495:         /* until the ".end" is found (or CPMEOF) */
496:         if ((fd = creat(filename)) == ERROR)
497:         {
498:             printf("\007\nCan't Create %s\n",filename);
499:             printf("Assuming out of directory or space\n");
500:             exit();

```

```

501:     }
502:     for(ptr2 = buffer2, n2 = BYTES_PER_BLOCK; n2--; )
503:         *ptr2++ = CPMEOF;
504:     ptr2 = buffer2;
505:     bytes_written = 0;
506:     rec_bytes = 0;
507:
508:     while((c=getabyte()) && c != CPMEOF)
509:     {
510:         if ((c == '.') && lookatend())
511:         {
512:             /* flush <cr,lf,"."> */
513:             while((c=getabyte()) &&
514:                 c != CPMEOF &&
515:                 (c == '\r' ||
516:                  c == '\n' ||
517:                  c == '.')) ;
518:             /* flush "end" line */
519:             while((c=getabyte()) &&
520:                 c != CPMEOF &&
521:                 c != '\r' &&
522:                 c != '\n') ;
523:             ungetbyte(c);
524:             break;
525:         }
526:         else
527:         {
528:             *ptr2++ = c;
529:             bytes_written++;
530:             if (bytes_written == BYTES_PER_BLOCK)
531:                 writebuf2(fd);
532:
533:             if (datasw)
534:             {
535:                 rec_bytes++;
536:                 if (rec_size == rec_bytes)
537:                 {
538:                     rec_bytes = 0;
539:                     if ((c=getabyte()) &&
540:                         !(c == '\n' || c == '\r'))
541:                         ungetbyte(c);
542:                     else if ((c=getabyte()) &&
543:                         !(c == '\n' || c == '\r'))
544:                         ungetbyte(c);
545:                     else if ((c=getabyte()) &&
546:                         !(c == '\n' || c == '\r'))
547:                         ungetbyte(c);
548:                     else if ((c=getabyte()) &&
549:                         !(c == '\n' || c == '\r'))
550:                         ungetbyte(c);
551:                     *ptr2++ = '\r';

```

```

551:         bytes_written++;
552:         if (bytes_written == BYTES_PER_BLOCK)
553:             writebuf2(fd);
554:         *ptr2++ = '\n';
555:         bytes_written++;
556:         if (bytes_written == BYTES_PER_BLOCK)
557:             writebuf2(fd);
558:     }
559: }
560: }
561: if (bytes_written) writebuf2(fd);
562: if (close(fd) == ERROR)
563: {
564:     printf("\nCan't Close File\007\n");
565:     exit();
566: }
567: }
568: /*
569: *****
570: *      lookatend    looks for '.end' statement
571: *****
572: */
573: int
574: lookatend()
575: {
576:     char work[11];
577:     char *wk;
578:     int w;
579:     for (wk = work, w = 11; w--; *wk++ = '\0') ;
580:     for (wk = work, w = 10; w-- ; *wk++ = getabyte()) ;
581:     for (w = 10; w-- ; wk-- , ungetbyte(*wk)) ;
582:     wk = work;
583:     while ((*wk == '\r') ||
584:            (*wk == '\n') ||
585:            (*wk == '.'))
586:         wk++;
587:     if (*wk++ == 'e' &&
588:        *wk++ == 'n' &&
589:        *wk++ == 'd' &&
590:        (*wk == '\r' || *wk == '\n'))
591:         return(1);
592:     else
593:         return(0);
594: }
595: /*
596: *****
597: *      getabyte     returns next byte from file buffer
598: *****
599: */
600: char
601: getabyte()

```

```

601: {
602:     char c;
603:     if (!bytes_read)
604:     {
605:         n=read(filein,buffer+l0,BLOCKS);
606:         bytes_read = (n * BYTES_PER_BLOCK);
607:         ptr = buffer+l0;
608:     }
609:     if (!bytes_read)
610:     {
611:         signal_eof++;
612:         return(CPMEOF);
613:     }
614:     bytes_read--;
615:     c = *ptr++;
616:     return(c);
617: /*
618: *****
619: *      ungetbyte    puts a byte back into the file buffer
620: *****
621: */
622: char
623: ungetbyte(c)
624: char c;
625: {
626:     bytes_read++;
627:     ptr--;
628:     *ptr = c;
629:     return(c);
630: }
631: /*
632: *****
633: *      writebuf2    writes output buffer2 and preps for next block
634: *****
635: */
636: int
637: writebuf2(fd)
638: int fd;
639: {
640:     int n2;
641:
642:     if (write(fd,buffer2,l) != l)
643:     {
644:         printf("\nWrite Error: %d\n",tell(fd));
645:         puts(errmsg(errno()));
646:         exit();
647:     }
648:     for(ptr2 = buffer2, n2 = BYTES_PER_BLOCK; n2--; )
649:         *ptr2++ = CPMEOF;
650:     ptr2 = buffer2;

```

```
651:         bytes_written = 0;
652:     }
653: /*
654: *****
655: * END OF SOURCE
656: *****
657: */
```

Characters: 16880

SECTION 2.9

File: INGRES.CMD

Size: 1K

Page: 1

```
1: * *****
2: * BOOTSTRAP FOR: INGRES MWT TEST
3: * this file is created dynamically by ingresdb.c
4: * *****
5: SET TALK OFF
6: ERASE
7: @ 10,14 SAY "DBASE INGRES MWT TEST"
8: STORE "MWT" TO DATABASE
9: STORE "TEST" TO RELATION
10: STORE " " TO INDEXER
11: DO INGRESMM
12: RETURN
```

Characters: 336

```

1: ***** INGRESMM.CMD *****
2: *
3: * INGRESMM.CMD MAIN MENU MODULE FOR DBASE II/INGRESS DEMO
4: *
5: * NOTE: THE GLOBAL VALUES "RELATION" AND "INDEXER" MUST BE
6: *     SET BEFORE THIS COMMAND STREAM IS EXECUTED
7: *****
8:
9: SET TALK OFF
10: STORE RELATION+".S" TO INGSTREAM
11: STORE RELATION+".ING" TO INGING
12: STORE RELATION+".DEF" TO INGDEF
13: STORE RELATION+".DAT" TO INGDAT
14: STORE RELATION+".DBF" TO INGDBF
15: STORE RELATION+".NDX" TO INGNDX
16: STORE T TO TRUE
17: DO WHILE TRUE
18:     ERASE
19:     SET COLON OFF
20:     SET CONSOLE ON
21: ? CHR(27)+CHR(70)
22: @ 3, 14 SAY "          faaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac"
23: @ 4, 14 SAY "          DBASE II/INGRES DEMO"
24: @ 5, 14 SAY "          eaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaad"
25: ? CHR(27)+CHR(71)
26: @ 7,14 SAY "      1> Load &RELATION from &INGDAT to DBASE II"
27: @ 8,14 SAY "      2> Display/Edit Records from TOP of file"
28: @ 9,14 SAY "      3> Display/Edit Records Indexed by &INDEXER"
29: @ 10,14 SAY "      4> Search/Edit/Delete by &INDEXER"
30: @ 11,14 SAY "      5> Add New &RELATION Database Records"
31: @ 12,14 SAY "      6> Display &RELATION Structure Format"
32: @ 13,14 SAY "      7> Rebuild &INGDAT for Reload to INGRES"
33: ? CHR(27)+CHR(70)
34: @ 15, 14 SAY "          aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
35: ? CHR(27)+CHR(71)
36: @ 16,14 SAY "      A> Shift to ANOTHER Database [DBASE II]"
37: @ 17,14 SAY "      C> Clear &RELATION Database [DBASE II]"
38: @ 18,14 SAY "      D> Delete &RELATION Database [DBASE II]"
39: @ 19,14 SAY "      E> Exit to DBASE II Prompt"
40: @ 20,14 SAY "      Q> Quit to CP/M Prompt"
41: @ 22,14 SAY "      Enter menu selection number or letter."
42:
43: *****
44: *     TRAP INVALID RESPONSES NOT ON MENU
45: *****
46: STORE T TO LEGAL
47: DO WHILE LEGAL
48:     SET CONSOLE OFF
49:     WAIT TO ANSWER
50:     SET CONSOLE ON

```

```
51:     STORE VAL(ANSWER) TO RANGE
52:     STORE !(ANSWER) TO ANSWER
53:     IF ANSWER="A" .OR. ANSWER="C" .OR. ANSWER="D" .OR. ANSWER="E" .OR. ANSWER="Q"
54:         STORE F TO LEGAL
55:     ENDIF
56:     IF LEGAL
57:         IF RANGE<1 .OR. RANGE>7
58:             LOOP
59:         ELSE
60:             STORE F TO LEGAL
61:         ENDIF Range
62:     ENDIF Legal
63: ENDDO Legal
64:
65:
66: *****
67: *     SUBSTITUTE ANOTHER RELATION FOR THE PRESENT ONE
68: *****
69:
70:     IF ANSWER="A"
71:         IF FILE("INGRESAN.CMD")
72:             DO INGRESAN
73:         ENDIF
74:     LOOP
75: ENDIF
76:
77:
78: *****
79: *     CLEAR DATA FROM &RELATION BY DELETE AND RE-CREATE
80: *****
81:
82:     IF ANSWER="C"
83:         IF FILE("INGRESCD.CMD")
84:             DO INGRESCD
85:         ENDIF
86:     LOOP
87: ENDIF
88:
89: *****
90: *     DELETE &RELATION FROM DISK, RETURN DISK SPACE TO SYSTEM
91: *****
92:
93:     IF ANSWER="D"
94:         IF FILE("INGRESDR.CMD")
95:             DO INGRESDR
96:         ENDIF
97:     LOOP
98: ENDIF
99:
100:
```

```
101: *****
102: *      EXIT TO DBASE II COMMAND LEVEL (ACTUALLY ANY HIGHER LEVEL)
103: *****
104:
105:      IF ANSWER="E"
106:          USE
107:          ERASE
108:          RETURN
109:      ENDIF Ans=e
110:
111:
112: *****
113: *      EXIT TO CP/M, COMPLETELY LEAVING DBASE II
114: *****
115:
116:      IF ANSWER="Q"
117:          USE
118:          ERASE
119:          QUIT
120:      ENDIF Ans=q
121:
122: *****
123: *      CREATE &RELATION AND APPEND DATA FROM &INGDAT
124: *****
125:
126:      IF ANSWER="1"
127:          IF FILE("INGRESCR.CMD")
128:              DO INGRESR
129:          ENDIF
130:      LOOP
131:  ENDIF
132:
133: *****
134: *      CALL TO EDIT MODULE FOR ANSWERS 2, 3, 4, 5
135: *****
136:
137:      IF RANGE>1 .AND. RANGE<6
138:          IF FILE("INGRESED.CMD")
139:              DO INGRESED
140:          ENDIF
141:      LOOP
142:  ENDIF
143:
144: *****
145: *      SHOW STRUCTURE OF &RELATION BY LISTING CONTENT OF INGRESDF
146: *****
147:
148:      IF ANSWER="6"
149:          IF FILE("INGRESSS.CMD")
150:              DO INGRESSS
```

```
151:         ENDIF
152:         LOOP
153:     ENDIF
154:
155:
156: *****
157: *         CALL &RELATION.DAT REBUILD TO GET READY TO RELOAD INGRES
158: *****
159:
160:     IF ANSWER="7"
161:         IF FILE("INGRESRB.CMD")
162:             DO INGRESRB
163:         ENDIF
164:     LOOP
165: ENDIF
166:
167:
168: ENDDO True
169: RETURN
```

Characters: 5066

SECTION 2.11

File: INGRESAN.CMD

Size: 2K

Page: 1

```

1: ***** INGRESAN.CMD *****
2: *
3: *      INGRESAN.CMD BRINGS IN ANOTHER 'DBF' AND BOOTS IT
4: *
5: *****
6:
7: SET TALK OFF
8: *
9: *****
10: * part 1 get relation name
11: *****
12: *
13: ERASE
14: STORE T TO LEGAL
15: DO WHILE LEGAL
16:     SET CONSOLE ON
17:     SET CONFIRM ON
18:     STORE "      " TO RELATION
19: @ 10,14 SAY "What is the name of the dBASE file?"
20: @ 11,14 SAY "(Use File Name Only, no extension.) ";
21: GET RELATION PICTURE "XXXXXXXX"
22:     SET CONSOLE OFF
23:     READ
24:     SET CONSOLE ON
25:     SET CONFIRM OFF
26:     STORE !(RELATION) TO RELATION
27:     STORE TRIM(RELATION) TO RELATION
28:     STORE RELATION+".DBF" TO INGSTREAM
29:     IF FILE(INGSTREAM)
30:         STORE F TO LEGAL
31:     ENDIF
32:
33: ENDDO
34: ERASE
35: STORE RELATION+".S" TO INGSTREAM
36: STORE RELATION+".ING" TO INGING
37: STORE RELATION+".DEF" TO INGDEF
38: STORE RELATION+".DAT" TO INGDAT
39: *
40: *****
41: * part 2 build <relation>.dat
42: *****
43: *
44: ERASE
45: @ 10,14 SAY "Standby while writing all records to &INGDAT"
46: USE &RELATION
47: SET TALK OFF
48: SET CONSOLE OFF
49: COPY TO &INGDAT SDF
50: *

```

```
51: *****
52: * part 2 build <relation>.def
53: *****
54: *
55: @ 11,14 SAY "Standby while writing structure to &INGDEF"
56: USE &RELATION
57: COPY TO INGRESDF STRUCTURE EXTENDED
58: USE INGRESDF
59: COPY TO &INGDEF SDF
60: *
61: *****
62: * part 3 exit to ingresul.c
63: *****
64: *
65: USE
66: @ 12,14 SAY "Standby while re-building &INGSTREAM"
67: @ 13,14 SAY "EXITING FROM DBASE (I WILL RETURN!)"
68: DELETE FILE &INGSTREAM
69: DELETE FILE &INGING
70: DELETE FILE &INGCMD
71: QUIT TO "INGRESUL &DATABASE &RELATION",;
72: "INGRESDB &DATABASE &RELATION",;
73: "DBASE INGRES"
```

Characters: 1766

SECTION 2.12

File: INGRES.D.CMD

Size: 2K

Page: 1

```
1: ***** INGRES.D.CMD *****
2: *      CLEAR DATA FROM &RELATION BY DELETE AND RE-CREATE
3: *****
4:
5: ERASE
6: SET TALK OFF
7: SET CONSOLE OFF
8: STORE F TO LEGAL
9: IF FILE("&INGDBF") .AND. FILE("INGRES.DF.DBF")
10:     STORE T TO LEGAL
11: ENDIF
12: IF LEGAL
13:     ERASE
14: @ 10,14 SAY 'Rebuilding Structure of &RELATION'
15:     USE INGRES.DF
16:     DELETE ALL
17:     PACK
18:     USE &RELATION
19:     COPY TO INGRES.DF STRUCTURE EXTENDED
20: @ 11,14 SAY 'Standby while clearing &RELATION Database.'
21:     USE
22:     DELETE FILE &INGDBF
23:     IF FILE("&INGNDX")
24:         DELETE FILE &INGNDX
25:     ENDIF answer
26: @ 12,14 SAY "Creating &RELATION Database (EMPTY)"
27:     CREATE &RELATION FROM INGRES.DF
28:     USE &RELATION
29:     IF INDEXER<>" "
30:         INDEX ON &INDEXER TO &RELATION
31:     ENDIF Indexer
32: ELSE
33: @ 10,14 SAY "This option requires &INGDBF"
34: @ 11,14 SAY "A non-existent DBASE Database file"
35: @ 12,14 SAY "Try using another DBASE database file"
36: @ 13,14 SAY "From the main menu. Press return to go."
37:     WAIT TO ANSWER
38: ENDIF
39: STORE T TO TRUE
40: STORE "N" TO ANSWER
41: RETURN
42:
```

Characters: 1094

SECTION 2.13

File: INGRESR.CMD

Size: 2K

Page: 1

```

1: ***** INGRESR.CMD *****
2: * INGRESR.CMD CREATES dBASE II ".DBF" AND APPENDS DATA
3: *****
4: ERASE
5: SET TALK OFF
6: SET CONSOLE OFF
7: STORE F TO LEGAL
8: IF FILE("&INGDAT") .AND. FILE("&INGDEF")
9:     STORE T TO LEGAL
10: ENDIF
11: IF LEGAL
12:     ERASE
13: @ 10,14 SAY 'Rebuilding Structure of &RELATION from &INGDEF'
14:     USE INGRESDF
15:     DELETE ALL
16:     PACK
17:     APPEND FROM &INGDEF SDF
18: @ 11,14 SAY 'Standby while clearing &RELATION Database.'
19:     DELETE FILE &RELATION
20: @ 12,14 SAY "Creating &RELATION Database with data from &INGDAT"
21:     CREATE &RELATION FROM INGRESDF
22:     USE &RELATION
23:     APPEND FROM &INGDAT SDF
24: *
25: * INDEX BY SPECIFIED FIELD OR ASK ABOUT INDEXING
26: *
27:     IF INDEXER<>" "
28: @ 13,14 SAY "Indexing &RELATION by &INDEXER"
29:     INDEX ON &INDEXER TO &RELATION
30:     ELSE
31:         STORE "any" TO ANSWER
32:         DO WHILE ANSWER<>"Y" .AND. ANSWER<>"N"
33:             STORE " " TO ANSWER
34: @ 14,14 SAY "Want INDEX created for &RELATION <y,n>?";
35: GET ANSWER PICTURE "xxx"
36:         SET COLON ON
37:         SET CONSOLE ON
38:         READ
39:         SET CONSOLE OFF
40:         SET COLON OFF
41:         STORE !(ANSWER) TO ANSWER
42:         STORE TRIM(ANSWER) TO ANSWER
43:         ENDDO answer
44:         IF ANSWER="Y" .AND. FILE("INGRESIX.CMD")
45:             DO INGRESIX
46:             STORE "N" TO ANSWER
47:         ENDIF y
48:     ENDIF Indexer
49: *
50: *

```

```
51: *
52: ELSE
53: @ 10,14 SAY "This option requires &INGDEF AND &INGDAT"
54:     IF FILE("&INGSTREAM")
55: @ 11,14 SAY "Stand by while building them from &INGSTREAM"
56: @ 12,14 SAY "PLEASE RE-TRY WHEN THE MAIN MENU RETURNS"
57:     QUIT TO "INGRESDB &DATABASE &RELATION &INDEXER","DBASE INGRES"
58:     ENDIF
59: @ 11,14 SAY "They cannot be re-built from &INGSTREAM"
60: @ 12,14 SAY "Try to use another DBASE database file"
61: @ 13,14 SAY "From the main menu. Press return to go."
62:     WAIT TO ANSWER
63: ENDIF
64: STORE "N" TO ANSWER
65: STORE T TO TRUE
66: STORE T TO LEGAL
67: RETURN
```

Characters: 1859

SECTION 2.14

File: INGRES.DR.CMD

Size: 1K

Page: 1

```
1: ***** INGRES.DR.CMD *****
2: * INGRES.DR.CMD WILL DELETE &RELATION FROM DBASE
3: *****
4:
5: ERASE
6: SET TALK OFF
7: SET CONSOLE OFF
8: STORE F TO LEGAL
9: IF FILE("&INGDBF")
10:     STORE T TO LEGAL
11: ENDIF
12: IF LEGAL
13:     ERASE
14:     USE INGRES.DF
15: @ 10,14 SAY 'Standby while DELETING &RELATION Database.'
16:     USE
17:     DELETE FILE &RELATION
18:     IF FILE("&INGNDX")
19:         DELETE FILE &INGNDX
20:     ENDIF answer
21: ELSE
22: @ 10,14 SAY "This option requires &INGDBF"
23: @ 11,14 SAY "A non-existent DBASE Database file"
24: @ 12,14 SAY "Try using another DBASE database file"
25: @ 13,14 SAY "From the main menu. Press return to go."
26:     WAIT TO ANSWER
27: ENDIF
28: STORE "N" TO ANSWER
29: STORE T TO TRUE
30: RETURN
31:
```

Characters: 761

SECTION 2.15

File: INGRESED.CMD

Size: 2K

Page: 1

```

1: ***** INGRESED.CMD *****
2: *
3: * INGRESED.CMD edit control procedure for dBASE II/INGRESS
4: *      demo program.
5: *
6: *****
7:
8: USE &RELATION
9: DO WHILE TRUE
10:   ERASE
11:   IF ANSWER="2"
12:     SET INDEX TO
13:     BROWSE
14:     STORE T TO TRUE
15:     RETURN
16:   ENDIF Ans=2
17:   IF ANSWER="3"
18:     SET INDEX TO &RELATION
19:     BROWSE
20:     STORE T TO TRUE
21:     RETURN
22:   ENDIF Ans=3
23:   IF ANSWER="4"
24:     SET INDEX TO &RELATION
25:     ERASE
26:     STORE T TO TRUE
27:     DO WHILE TRUE
28:       STORE " " TO NM
29: @ 10,14 SAY ' Enter name of individual ' GET NM PICTURE "AAAAAAAAAAAAAA"
30:       SET CONSOLE OFF
31:       READ
32:       SET CONSOLE ON
33:       FIND &NM
34:       IF #=0
35:         ERASE
36: @ 10,14 SAY ' No find - do you wish to try again? (Y/N)'
37:       SET CONSOLE OFF
38:       WAIT TO ANSWER
39:       SET CONSOLE ON
40:       IF !(ANSWER)="Y"
41:         LOOP
42:       ELSE
43:         STORE F TO TRUE
44:       ENDIF Ans=y
45:     ELSE
46:       BROWSE
47:       ERASE
48: @ 10,14 SAY ' Are you ready to Pack database? (Y/N)'
49:       SET CONSOLE OFF
50:       WAIT TO ANSWER

```

```
51:          SET CONSOLE ON
52:          IF !(ANSWER)="Y"
53: @ 10,0
54: @ 10,14 SAY 'Standby while packing database for deleted files.'
55:          PACK
56:          STORE T TO TRUE
57:          RETURN
58:        ELSE
59:          STORE T TO TRUE
60:          RETURN
61:      ENDIF
62:  ENDIF #=0
63:  ENDDO True
64:  STORE T TO TRUE
65:  RETURN
66:  ENDIF Ans=4
67:  IF ANSWER="5"
68:    SET INDEX TO &RELATION
69:    APPEND BLANK
70:    BROWSE
71:    STORE T TO TRUE
72:    RETURN
73:  ENDIF Ans=5
74: ENDDO True
75: STORE T TO TRUE
76: STORE "N" TO ANSWER
77: RETURN
```

Characters: 1861

SECTION 2.16

File: INGRESIX.CMD

Size: 3K

Page: 1

```

1: ***** INGRESIX.CMD *****
2: * INGRESIX.CMD CREATES dBASE II ".NDX" FILE FOR &RELATION
3: *****
4: ERASE
5: SET TALK OFF
6: STORE F TO LEGAL
7: IF FILE("&INGDBF") .AND. FILE("INGRESDF.DBF")
8:     STORE T TO LEGAL
9: ENDIF
10: IF .NOT. LEGAL
11: @ 1,1 SAY "The files &INGDBF and INGRESDF.DBF are needed"
12: @ 2,1 SAY "dBase II failure in INGRESIX.CMD"
13: RETURN
14: ENDIF
15: *
16: *****
17: * part 1 display field names
18: *****
19: *
20: ERASE
21: @ 1,1 SAY 'PLEASE SELECT INDEX FIELD NAME'
22:     STORE 3 TO W
23:     USE INGRESDF
24:     GO TOP
25:     STORE T TO TRUE
26:     DO WHILE TRUE
27:         IF .NOT. EOF
28: @ W,1 SAY ">" + FIELD:NAME
29:             SKIP
30:         ENDIF
31:         IF .NOT. EOF
32: @ W,21 SAY ">" + FIELD:NAME
33:             SKIP
34:         ENDIF
35:         IF .NOT. EOF
36: @ W,41 SAY ">" + FIELD:NAME
37:             SKIP
38:         ENDIF
39:         IF .NOT. EOF
40: @ W,61 SAY ">" + FIELD:NAME
41:             SKIP
42:         ENDIF
43:         IF EOF
44:             STORE F TO TRUE
45:         ENDIF
46:         STORE W+1 TO W
47:     ENDDO true
48: *
49: *****
50: * part 2 enter selection

```

```
51: *****
52: *
53:     STORE T TO TRUE
54:     DO WHILE TRUE
55:         SET COLON ON
56:         SET CONSOLE ON
57:         SET CONFIRM ON
58:         STORE " " TO INDEXER
59: @ 1,32 SAY ">";
60: GET INDEXER PICTURE "XXXXXXXXXX"
61:     SET CONSOLE OFF
62:     READ
63:     SET CONSOLE ON
64:     SET CONFIRM OFF
65:     SET COLON OFF
66:     STORE !(INDEXER) TO INDEXER
67:     STORE TRIM(INDEXER) TO INDEXER
68: @ 1,33 SAY "&INDEXER"
69: *
70: *****
71: * part 3 validate selection
72: *****
73: *
74:     USE INGRESDF
75:     GO TOP
76:     DO WHILE TRUE
77:         IF .NOT. EOF
78:             STORE FIELD:NAME TO ANSWER
79:             STORE !(ANSWER) TO ANSWER
80:             STORE TRIM(ANSWER) TO ANSWER
81:             IF "&INDEXER" = "&ANSWER"
82:                 STORE F TO TRUE
83:             ELSE
84:                 SKIP
85:             ENDIF
86:         ELSE
87:             STORE F TO TRUE
88:         ENDIF
89:     ENDDO true
90:     IF EOF
91: @ 1,44 SAY "NAME INVALID"
92:         GO TOP
93:         STORE T TO TRUE
94:         STORE 25 TO W
95:         DO WHILE W<>0
96:             STORE W-1 TO W
97:         ENDDO w
98: @ 1,33 SAY "
99: @ 1,33 SAY ">"
100:     ENDIF eof
```

```
101:          ENDDO true
102: *
103: *****
104: * part 4 index by selection
105: *****
106: *
107: @ 1,33 SAY "Indexing &RELATION by &INDEXER"
108:          USE &RELATION
109:          INDEX ON &INDEXER TO &RELATION
110: STORE "N" TO ANSWER
111: STORE T TO TRUE
112: STORE T TO LEGAL
113: RELEASE W
114: RETURN
```

Characters: 2334

SECTION 2.17

File: INGRESRB.CMD

Size: 1K

Page: 1

```

1: ***** INGRESRB.CMD *****
2: *
3: *      INGRESRB.CMD CREATES some FILES FOR UPLOADING TO INGRES
4: *      FROM EXISTING DBASE <&RELATION>.DBF
5: *
6: *****
7:
8: *
9: *****
10: * part 1 build <relation>.dat
11: *****
12: *
13: ERASE
14: @ 10,14 SAY "Standby while writing all records to &INGDAT"
15: USE &RELATION
16: SET TALK OFF
17: SET CONSOLE OFF
18: COPY TO &INGDAT SDF
19: *
20: *****
21: * part 2 build <relation>.def
22: *****
23: *
24: @ 11,14 SAY "Standby while writing structure to &INGDEF"
25: USE &RELATION
26: COPY TO INGRESDF STRUCTURE EXTENDED
27: USE INGRESDF
28: COPY TO &INGDEF SDF
29: *
30: *****
31: * part 3 exit to ingresul.c
32: *****
33: *
34: USE
35: @ 12,14 SAY "Standby while re-building &INGSTREAM"
36: @ 13,14 SAY "EXITING FROM DBASE (I WILL RETURN!)"
37: DELETE FILE &INGING
38: QUIT TO "INGRESUL &DATABASE &RELATION", "DBASE INGRES"

```

Characters: 1012

SECTION 2.18

File: INGRESSS.CMD

Size: 1K

Page: 1

```
1: ***** INGRESSS.CMD *****
2: * INGRESSS.CMD WILL SHOW STRUCTURE OF &RELATION FROM DBASE
3: *****
4:
5: ERASE
6: SET TALK OFF
7: SET CONSOLE OFF
8: STORE F TO LEGAL
9: IF FILE("&INGDBF")
10:     STORE T TO LEGAL
11: ENDIF
12: IF LEGAL
13: @ 1,1 SAY 'Structure of &RELATION'
14:     USE INGRESDF
15:     DELETE ALL
16:     PACK
17:     USE &RELATION
18:     COPY TO INGRESDF STRUCTURE EXTENDED
19:     USE INGRESDF
20:     SET CONSOLE ON
21:     LIST
22: @ 1,40 SAY '..Press Return to Continue..'
23:     SET CONSOLE OFF
24:     WAIT TO ANSWER
25: ELSE
26: @ 10,14 SAY "This option requires &INGDBF"
27: @ 11,14 SAY "A non-existant DBASE Database file"
28: @ 12,14 SAY "Try using another DBASE database file"
29: @ 13,14 SAY "From the main menu. Press return to go."
30:     WAIT TO ANSWER
31: ENDIF
32: STORE "N" TO ANSWER
33: STORE T TO TRUE
34: RETURN
```

Characters: 845

SECTION 3

SECTION 3.1

MAKEFILE

```
# This requires you to be at the SUPERUSER level for installation.
# Also you must be in the directory that contains Makefile.
# To execute: make install          --will install the following
# or
# To recompile supert: make supert  --will recompile super
#
install:
    cp bin/super /usr/local
    chmod 0755 /usr/local/super
    chown ty /usr/local/super
#
    cp bin/superd /usr/local
    chmod 0755 /usr/local/superd
    chown ty /usr/local/superd
#
    cp bin/supert /usr/local/lib/scomp/bin
    chmod 0755 /usr/local/lib/scomp/bin/supert
    chown ty /usr/local/lib/scomp/bin/supert
#
    cp man/super.1 /usr/man/man1
    chown ty /usr/man/man1/super.1
    man super >temp
#
    cp man/superd.1 /usr/man/man1
    chown ty /usr/man/man1/superd.1
    man superd >temp
#
    cp man/supert.1 /usr/man/man1
    chown ty /usr/man/man1/supert.1
    man supert >temp
#
    rm temp

supert: src/supert.c
    cc src/supert.c -O -o bin/supert
```

SECTION 3.2

super(1L)

UNIX Programmer's Manual

super(1L)

NAME

super - SUPERCOMP-TWENTY (C)

SYNOPSIS

super [<command-file>]

DESCRIPTION

The super command is used to start the SUPERCOMP-TWENTY electronic spread sheet package. You may optionally specify <command-file> to identify a file of SUPERCOMP-TWENTY commands to be executed when the program is started.

A summary of the SUPERCOMP-TWENTY commands:

<u>BACKSPACE</u>	Aborts an entry or command (before pressing return).
<u>HELP</u>	Displays Help Text. [Note: usually "f2" key]
<u>SPACE BAR</u>	Recalculates the worksheet.
<u>PAGE</u>	Scr .

SECTION 3.3

SUPER NROFF FILE

.TH super 1L NOSC
.UC 4
.SH NAME
super \- SUPERCOMP-TWENTY (C)
.SH SYNOPSIS
.B super
.RB [<command-file>]
.SH DESCRIPTION
The
.I super
command is used to start the SUPERCOMP-TWENTY electronic
spread sheet package. You may optionally specify
.I <command-file>
to identify a file of SUPERCOMP-TWENTY commands to be executed when the
program is started.
.sp
A summary of the SUPERCOMP-TWENTY commands:
.TP 1.2i
.I BACKSPACE
Aborts an entry or command (before pressing return).
.TP
.I HELP
Displays Help Text. [Note: usually "f2" key]
.TP
.I SPACE BAR
Recalculates the worksheet.
.TP
.I PAGE
Scrolls the worksheet in the direction of the arrow key
pressed after it. [Note: usually "f3" key]
.TP
.I GOTO (>)
Allows you to move the cursor directly to a specified cell.
[Note: usually "f4" key]
.TP
.I /
First character of every keyboard command sequence.
.TP
.I B
Blank a range of cells.
.TP
.I C
Clear the worksheet.
.TP
.I D
Delete current column or row
.TP
.I E
Edit a formula in a cell.
.TP
.I F
Format display style for a cell.
.TP
.I I
Insert column or row.
.TP
.I M

Move the current row or column to a new location.
 .TP
 .I O
 Set Options for Printer, Terminal, Delimiter.
 .TP
 .I P
 Print all or part of a worksheet.
 .TP
 .I R
 Replicate cells from source into new location.
 .TP
 .I S
 Storage (file) commands.
 .TP
 .I T
 Title set up commands.
 .TP
 .I W
 Window set up commands.
 .TP
 .I X
 Exit from SUPERCOMP-TWENTY
 .SH NOTE
 SUPERCOMP-TWENTY will ignore most traditional UNIX keyboard "interrupt"
 and "quit" signals. To terminate the package you must use the
 .I /XY
 or
 .I /XS
 commands.
 .SH FILES
 All files created by SUPERCOMP will be directed to the
 .I current working directory.
 .SH DIAGNOSTICS
 The messages issued by SUPERCOMP-TWENTY are explained in the
 SUPERCOMP-TWENTY HANDBOOK.
 .SH "SEE ALSO"
 superd(1L), supert(1L)
 .SH AUTHOR
 SUPERCOMP-TWENTY (C) is a product of Access Technology, Inc.

SECTION 3.4

superd(1L)

UNIX Programmer's Manual

superd(1L)

NAME

superd - SUPERCOMP-TWENTY (C) Demo package

SYNOPSIS

superd [<command-file>]

DESCRIPTION

The superd command is used to start the SUPERCOMP-TWENTY electronic spread sheet package. You may optionally specify <command-file> to identify a file of SUPERCOMP-TWENTY commands to be executed when the program is started.

A summary of the SUPERCOMP-TWENTY commands:

<u>BACKSPACE</u>	Aborts an entry or command (before pressing return).
<u>HELP</u>	Displays Help Text. [Note: usually "f2" key]
<u>SPACE BAR</u>	Recalculates the worksheet.
<u>PAGE</u>	Scrolls the worksheet in the direction of the arrow key pressed after it. [Note: usually "f3" key]
<u>GOTO</u> (>)	Allows you to move the cursor directly to a specified cell. [Note: usually "f4" key]
/	First character of every keyboard command sequence.
<u>B</u>	Blank a range of cells.
<u>C</u>	Clear the worksheet.
<u>D</u>	Delete current column or row
<u>E</u>	Edit a formula in a cell.
<u>F</u>	Format display style for a cell.
<u>I</u>	Insert column or row.
<u>M</u>	Move the current row or column to a new location.

<u>O</u>	Set Options for Printer, Terminal, Delimiter.
<u>P</u>	Print all or part of a worksheet.
<u>R</u>	Replicate cells from source into new location.
<u>S</u>	Storage (file) commands.
<u>T</u>	Title set up commands.
<u>W</u>	Window set up commands.
<u>X</u>	Exit from SUPERCOMP-TWENTY

NOTE

SUPERCOMP-TWENTY will ignore most traditional UNIX keyboard "interrupt" and "quit" signals. To terminate the package you must use the /XY or /XS commands.

FILES

All files created by SUPERCOMP will be directed to a special subdirectory of the current working directory.

DIAGNOSTICS

The messages issued by SUPERCOMP-TWENTY are explained in the

SEE ALSO

super(1L), supert(1L) SUPERCOMP-TWENTY HANDBOOK.

AUTHOR

SUPERCOMP-TWENTY (C) is a product of Access Technology, Inc.

SECTION 3.5

SUPERD NROFF FILE

```
.TH superd 1L NOSC
.UC 4
.SH NAME
superd \- SUPERCOMP-TWENTY (C) Demo package
.SH SYNOPSIS
.B superd
.RB [<command-file>]
.SH DESCRIPTION
The
.I superd
command is used to start the SUPERCOMP-TWENTY electronic
spread sheet package. You may optionally specify
.I <command-file>
to identify a file of SUPERCOMP-TWENTY commands to be executed when the
program is started.
.SP
A summary of the SUPERCOMP-TWENTY commands:
.TP 1.2i
.I BACKSPACE
Aborts an entry or command (before pressing return).
.TP
.I HELP
Displays Help Text. [Note: usually "f2" key]
.TP
.I SPACE BAR
Recalculates the worksheet.
.TP
.I PAGE
Scrolls the worksheet in the direction of the arrow key
pressed after it. [Note: usually "f3" key]
.TP
.I GOTO (>)
Allows you to move the cursor directly to a specified cell.
[Note: usually "f4" key]
.TP
.I /
First character of every keyboard command sequence.
.TP
.I B
Blank a range of cells.
.TP
.I C
Clear the worksheet.
.TP
.I D
Delete current column or row
.TP
.I E
Edit a formula in a cell.
.TP
.I F
Format display style for a cell.
```

.TP
.I I
Insert column or row.
.TP
.I M
Move the current row or column to a new location.
.TP
.I O
Set Options for Printer, Terminal, Delimiter.
.TP
.I P
Print all or part of a worksheet.
.TP
.I R
Replicate cells from source into new location.
.TP
.I S
Storage (file) commands.
.TP
.I T
Title set up commands.
.TP
.I W
Window set up commands.
.TP
.I X
Exit from SUPERCOMP-TWENTY
.SH NOTE
SUPERCOMP-TWENTY will ignore most traditional UNIX keyboard "interrupt"
and "quit" signals. To terminate the package you must use the
.I /XY
or
.I /XS
commands.
.SH FILES
All files created by SUPERCOMP will be directed to a special subdirectory of the
.I current working directory.
.SH DIAGNOSTICS
The messages issued by SUPERCOMP-TWENTY are explained in the
.SH "SEE ALSO"
super(1L), supert(1L)
SUPERCOMP-TWENTY HANDBOOK.
.SH AUTHOR
SUPERCOMP-TWENTY (C) is a product of Access Technology, Inc.

SECTION 3.6

supert(1L)

UNIX Programmer's Manual

supert(1L)

NAME

supert - create terminal description file for SUPERCOMP-TWENTY (C)

SYNOPSIS

supert

DESCRIPTION

The supert command will create a terminal description file in the format required by the SUPERCOMP-TWENTY electronic spread sheet package. The current value of the TERMCAP string in the users environment is used to build the file.

The name of the file will be .supertfil in the current working directory. You may customize this file to set up your own terminal graphics characteristics. Customization notes are being prepared and will be made available here as time permits.

NOTE

The commands super and superd will look for the terminal description file .supertfil in your home directory. If not found in your home directory, SUPERCOMP-TWENTY will use its own terminal description file.

DIAGNOSTICS

Messages issued by supert will warn of any illogical conditions found in the TERMCAP string.

SEE ALSO

super(1L), superd(1L), termcap(5)

AUTHOR

Mike Trest, System Development Corporation

**NAVAL OCEAN SYSTEMS CENTER, SAN DIEGO, CA
INGRES TO DBASE AND FINANCIAL SPREADSHEET
PROGRAM PACKAGES BY J BAIRD SYSTEM
DEVELOPMENT CORPORATION**

**NOSC CR 223
UNCLASSIFIED
FEB 1984**

END
DATE
FILMED
Mar 15 194

SECTION 3.7

SUPERT NROFF FILE

```
.TH supert 1L NOSC
.UC 4
.SH NAME
supert \- create terminal description file for SUPERCOMP-TWENTY (C)
.SH SYNOPSIS
.B supert
.SH DESCRIPTION
The
.I supert
command will create a terminal description file in the format
required by the SUPERCOMP-TWENTY electronic
spread sheet package. The current value of the TERMCAP string in
the users environment is used to build the file.
.sp
The name of the file will be
.I .supertfil
in the current working directory. You may customize this file
to set up your own terminal graphics characteristics.
Customization notes are being prepared and will be made available
here as time permits.
.SH NOTE
The commands
.I super
and
.I superd
will look for the terminal description file
.I .supertfil
in your
.I home directory.
If not found in your home directory, SUPERCOMP-TWENTY will use
its own terminal description file.
.SH DIAGNOSTICS
Messages issued by
.I supert
will warn of any illogical conditions found in the TERMCAP
string.
.SH "SEE ALSO"
super(1L), superd(1L), termcap(5)
.SH AUTHOR
Mike Trest, System Development Corporation
```

SECTION 3.6

supert(1L)

UNIX Programmer's Manual

supert(1L)

NAME

supert - create terminal description file for SUPERCOMP-TWENTY (C)

SYNOPSIS

supert

DESCRIPTION

The supert command will create a terminal description file in the format required by the SUPERCOMP-TWENTY electronic spread sheet package. The current value of the TERMCAP string in the users environment is used to build the file.

The name of the file will be .supertfil in the current working directory. You may customize this file to set up your own terminal graphics characteristics. Customization notes are being prepared and will be made available here as time permits.

NOTE

The commands super and superd will look for the terminal description file .supertfil in your home directory. If not found in your home directory, SUPERCOMP-TWENTY will use its own terminal description file.

DIAGNOSTICS

Messages issued by supert will warn of any illogical conditions found in the TERMCAP string.

SEE ALSO

super(1L), superd(1L), termcap(5)

AUTHOR

Mike Trest, System Development Corporation

SECTION 3.8

File: SUPERT.C Size: 16K

Page: 1

```
1: /*
2: *****
3: * superterm.c create SUPERCOMP-TWENTY "TFIL" from TERMCAP environment
4: *****
5: */
6: #include <stdio.h>
7: #include <ctype.h>
8:
9: #define TFIL ".supertfil"
10: static char od[]="01234567";
11: FILE *fpa;
12: char *eptr;
13: char *tptr;
14: char *here;
15:
16: char *getenv();
17: char *index();
18: char *finditem();
19:
20: int kn[20];
21: int i,j,k;
22:
23: char tfil[50];
24: char term[50];
25: char value[100];
26: char padvalue[20];
27: char padoctal[20];
28: char line[100];
29:
30: char ku[40];
31: char kd[40];
32: char kr[40];
33: char kl[40];
34: char khelp[20];
35: char kpage[20];
36: char kgoto[20];
37:
38: main(argc,argv)
39: int argc;
40: char *argv[];
41: {
42:
43: /* open file for this program */
44:     strcpy(tfil,TFIL);
45:     if ((fpa = fopen(tfil,"w+")) == NULL)
46:     {
47:         printf("\nCANNOT CREATE %s\n",tfil);
48:         exit(-1);
49:     }
50:
```

```

51: /* get pointer to TERM and TERMCAP */
52:     if ((eptr = getenv("TERM")) == NULL)
53:     {
54:         printf("\ncannot find TERMCAP in environment\n");
55:         close(fpa);
56:         exit(-1);
57:     }
58:     else strcpy(term,eptr);
59:
60:     if ((eptr = getenv("TERMCAP")) == NULL)
61:     {
62:         printf("\ncannot find TERMCAP in environment\n");
63:         close(fpa);
64:         exit(-1);
65:     }
66:
67:
68: /*
69:  * heading line of the file
70:  */
71:     fprintf(fpa,"%!s ",term);
72:     *line = '\0';
73:     if ((finditem("li#",eptr,value)) != NULL)
74:         strcat(line,value ? padvalue : value);
75:     else    strcat(line,"24");
76:     strcat(line," ");
77:     if ((finditem("co#",eptr,value)) != NULL)
78:         strcat(line,value ? padvalue : value);
79:     else    strcat(line,"80");
80:     fprintf(fpa,"%s %s\n",line,term);
81:
82: /*
83:  * INPUT keyboard sequence: LF, CR
84:  */
85:     fprintf(fpa," INPUT = \\12\\200\\200",value);
86:     if ((finditem("cr=",eptr,value)) != NULL)
87:     {
88:         fprintf(fpa,"%s\\200\\200\n",value);
89:     }
90:     else    fprintf(fpa,"\\15\\200\\200\n");
91:
92: /*
93:  * INPUT keyboard sequence: UP arrow
94:  */
95:     if ((finditem("ku=",eptr,value)) != NULL)
96:     {
97:         strcpy(ku,value);
98:         fprintf(fpa," INPUT = %s\\200\\1\n",ku);
99:     }
100:    else if ((finditem("up=",eptr,value)) != NULL)

```

```

101:      {
102:          strcpy(ku,value);
103:          fprintf(fpa," INPUT = %s\\200\\1\\n",ku);
104:      }
105:
106: /*
107:  * INPUT keyboard sequence: DOWN arrow
108:  */
109:      if ((finditem("kd=",eptr,value)) != NULL)
110:      {
111:          strcpy(kd,value);
112:          fprintf(fpa," INPUT = %s\\200\\2\\n",kd);
113:      }
114:      else if ((finditem("do=",eptr,value)) != NULL)
115:      {
116:          strcpy(kd,value);
117:          fprintf(fpa," INPUT = %s\\200\\2\\n",kd);
118:      }
119:
120: /*
121:  * INPUT keyboard sequence: RIGHT arrow
122:  */
123:      if ((finditem("kr=",eptr,value)) != NULL)
124:      {
125:          strcpy(kr,value);
126:          fprintf(fpa," INPUT = %s\\200\\3\\n",kr);
127:      }
128:
129: /*
130:  * INPUT keyboard sequence: LEFT arrow
131:  */
132:      if ((finditem("kl=",eptr,value)) != NULL)
133:      {
134:          strcpy(kl,value);
135:          fprintf(fpa," INPUT = %s\\200\\4\\n",kl);
136:      }
137:      else if ((finditem("kb=",eptr,value)) != NULL)
138:      {
139:          strcpy(kl,value);
140:          fprintf(fpa," INPUT = %s\\200\\4\\n",kl);
141:      }
142:
143: /*
144:  * INPUT SPECIAL FUNCTION KEYS
145:  */
146:      strcpy(khelp,"\\33\\62");
147:      strcpy(kpage,"\\33\\63");
148:      strcpy(kgoto,"\\33\\64");
149:      if ((finditem("kn#",eptr,value)) != NULL)
150:      {

```

```

151:      kn[0] = atoi(value ? padvalue : value);
152:      if ((finditem("k2=",eptr,value)) != NULL)
153:      {
154:          strcpy(khelp,value);
155:          kn[2] = 2;
156:          fprintf(fpa," INPUT = %s\\200\\15\\n",khelp);
157:      }
158:      else printf("\\nBAD TERMCAP VALUE FOR 'k2'\\n");
159:      if ((finditem("k3=",eptr,value)) != NULL)
160:      {
161:          strcpy(kpage,value);
162:          kn[3] = 3;
163:          fprintf(fpa," INPUT = %s\\200\\11\\n",kpage,ku);
164:          fprintf(fpa," INPUT = %s\\200\\12\\n",kpage,kd);
165:          fprintf(fpa," INPUT = %s\\200\\13\\n",kpage,kr);
166:          fprintf(fpa," INPUT = %s\\200\\14\\n",kpage,kl);
167:          fprintf(fpa," INPUT = %s\\200\\377\\n", kpage,kpage);
168:      }
169:      else printf("\\nBAD TERMCAP VALUE FOR 'k3'\\n");
170:      if (kn[0] >= 4)
171:      {
172:          if ((finditem("k4=",eptr,value)) != NULL)
173:          {
174:              strcpy(kgoto,value);
175:              kn[4] = 4;
176:              fprintf(fpa," INPUT = %s\\200\\076\\n",kgoto);
177:          }
178:          else printf("\\nBAD TERMCAP VALUE FOR 'k4'\\n");
179:      }
180:      else
181:      {
182:          if ((finditem("k1=",eptr,value)) != NULL)
183:          {
184:              strcpy(kgoto,value);
185:              kn[1] = 1;
186:              fprintf(fpa," INPUT = %s\\200\\076\\n",kgoto);
187:          }
188:          else printf("\\nBAD TERMCAP VALUE FOR 'k1'\\n");
189:      }
190:      if (!kn[1] && (finditem("k1=",eptr,value)) != NULL)
191:      {
192:          fprintf(fpa," INPUT = %s\\200\\15\\n",value);
193:      }
194:      if (!kn[4] && (finditem("k4=",eptr,value)) != NULL)
195:      {
196:          fprintf(fpa," INPUT = %s\\200\\15\\n",value);
197:      }
198:      if (!kn[5] && (finditem("k5=",eptr,value)) != NULL)
199:      {
200:          fprintf(fpa," INPUT = %s\\200\\15\\n",value);

```

```

201:         }
202:         if (!kn[6] && (finditem("k6=",eptr,value)) != NULL)
203:         {
204:             fprintf(fpa," INPUT = %s\\200\\15\n",value);
205:         }
206:         if (!kn[7] && (finditem("k7=",eptr,value)) != NULL)
207:         {
208:             fprintf(fpa," INPUT = %s\\200\\15\n",value);
209:         }
210:         if (!kn[8] && (finditem("k8=",eptr,value)) != NULL)
211:         {
212:             fprintf(fpa," INPUT = %s\\200\\15\n",value);
213:         }
214:         if (!kn[9] && (finditem("k9=",eptr,value)) != NULL)
215:         {
216:             fprintf(fpa," INPUT = %s\\200\\15\n",value);
217:         }
218:         if (!kn[10] && (finditem("k10=",eptr,value)) != NULL)
219:         {
220:             fprintf(fpa," INPUT = %s\\200\\15\n",value);
221:         }
222:         if (!kn[11] && (finditem("k11=",eptr,value)) != NULL)
223:         {
224:             fprintf(fpa," INPUT = %s\\200\\15\n",value);
225:         }
226:         if (!kn[12] && (finditem("k12=",eptr,value)) != NULL)
227:         {
228:             fprintf(fpa," INPUT = %s\\200\\15\n",value);
229:         }
230:     }
231:     else
232:     {
233:         fprintf(fpa," INPUT = %s%s\\200\\11\n",kpage,ku);
234:         fprintf(fpa," INPUT = %s%s\\200\\12\n",kpage,kd);
235:         fprintf(fpa," INPUT = %s%s\\200\\13\n",kpage,kr);
236:         fprintf(fpa," INPUT = %s%s\\200\\14\n",kpage,kl);
237:         fprintf(fpa," INPUT = %s%s\\200\\377\n", kpage,kpage);
238:         fprintf(fpa,";\n;info: no TERMCAP function keys\n");
239:         fprintf(fpa,";\n;info: Please use following manual sequences:\n");
240:         fprintf(fpa,";\n;info:  'HELP' = ESC 2\n");
241:         fprintf(fpa,";\n;info:  'PAGE' = ESC 3\n");
242:         fprintf(fpa,";\n;info:  'GOTO' = ESC 4\n;\n");
243:     }
244:
245: /*
246:  * home cursor command
247:  */
248:     if ((finditem("ho=",eptr,value)) != NULL)
249:     {
250:         fprintf(fpa,"         ho = %s",value);

```

```
251:         if (*padoctal)
252:             fprintf(fpa,"\\207\\%s\\n",padoctal);
253:         else
254:             fprintf(fpa,"\\n");
255:     }
256: /*
257:  * erase to end of line
258:  */
259:     if ((finditem("ce=",eptr,value)) != NULL)
260:     {
261:         fprintf(fpa,"         el = %s",value);
262:         if (*padoctal)
263:             fprintf(fpa,"\\207\\%s\\n",padoctal);
264:         else
265:             fprintf(fpa,"\\207\\3\\n");
266:     }
267: /*
268:  * erase to end of screen
269:  */
270:     if ((finditem("cd=",eptr,value)) != NULL)
271:     {
272:         fprintf(fpa,"         es = %s",value);
273:         if (*padoctal)
274:             fprintf(fpa,"\\207\\%s\\n",padoctal);
275:         else
276:             fprintf(fpa,"\\207\\12\\n");
277:     }
278: /*
279:  * move cursor right
280:  */
281:     if ((finditem("nd=",eptr,value)) != NULL)
282:     {
283:         fprintf(fpa,"         ri = %s",value);
284:         if (*padoctal)
285:             fprintf(fpa,"\\207\\%s\\n",padoctal);
286:         else
287:             fprintf(fpa,"\\n");
288:     }
289: /*
290:  * move cursor to start of new line
291:  */
292:     if ((finditem("cr=",eptr,value)) != NULL)
293:     {
294:         fprintf(fpa,"         nl = %s",value);
295:         if (*padoctal)
296:             fprintf(fpa,"\\207\\%s\\n",padoctal);
297:         else
298:             fprintf(fpa,"\\n");
299:     }
300:     else
301:         fprintf(fpa,"         nl = \\15\\12\\n");
```

```

301: /*
302:  * forward scroll screen by one line when at bottom
303:  */
304:     if ((finditem("sf=",eptr,value)) != NULL)
305:     {
306:         fprintf(fpa,"          fs = %s",value);
307:         if (*padoctal)
308:             fprintf(fpa,"\\207\\%s\\n",padoctal);
309:         else
310:             fprintf(fpa,"\\n");
311:     }
312:     else if ((finditem("cr=",eptr,value)) != NULL)
313:     {
314:         fprintf(fpa,"          fs = %s",value);
315:         if (*padoctal)
316:             fprintf(fpa,"\\207\\%s\\n",padoctal);
317:         else
318:             fprintf(fpa,"\\n");
319:     }
320: /*
321:  * reverse scroll screen by one line
322:  */
323:     if ((finditem("sr=",eptr,value)) != NULL)
324:     {
325:         fprintf(fpa,"          rs = %s",value);
326:         if (*padoctal)
327:             fprintf(fpa,"\\207\\%s\\n",padoctal);
328:         else
329:             fprintf(fpa,"\\n");
330:     }
331: /*
332:  * x,y cursor movements
333:  */
334:     if ((finditem("cm=",eptr,value)) != NULL)
335:     {
336:         transxy(value);
337:         fprintf(fpa,"          dc = %s",value);
338:         if (*padoctal)
339:             fprintf(fpa,"\\207\\%s\\n",padoctal);
340:         else
341:             fprintf(fpa,"\\207\\10\\n");
342:     }
343: /*
344:  * set up graphic options
345:  */
346:     if ((finditem("as=",eptr,value)) != NULL)
347:     {
348:         fprintf(fpa,"          gl = %s\\200a",value);
349:         if (*padoctal)
350:             fprintf(fpa,"\\207\\%s\\n",padoctal);
351:         else
352:             fprintf(fpa,"\\n");

```

```

351:         fprintf(fpa, "          g2 = %s\\200a",value);
352:         if (*padoctal)
353:             fprintf(fpa, "\\207\\ %s\\n",padoctal);
354:         else fprintf(fpa, "\\n");
355:         fprintf(fpa, "          g3 = %s\\200a",value);
356:         if (*padoctal)
357:             fprintf(fpa, "\\207\\ %s\\n",padoctal);
358:         else fprintf(fpa, "\\n");
359:         fprintf(fpa, "          g4 = %s\\200a",value);
360:         if (*padoctal)
361:             fprintf(fpa, "\\207\\ %s\\n",padoctal);
362:         else fprintf(fpa, "\\n");
363:         fprintf(fpa, "          g5 = %s\\200a",value);
364:         if (*padoctal)
365:             fprintf(fpa, "\\207\\ %s\\n",padoctal);
366:         else fprintf(fpa, "\\n");
367:         fprintf(fpa, "          g6 = %s\\200a",value);
368:         if (*padoctal)
369:             fprintf(fpa, "\\207\\ %s\\n",padoctal);
370:         else fprintf(fpa, "\\n");
371:         if ((finditem("ae=",eptr,value)) != NULL)
372:         {
373:             fprintf(fpa, "          gf = %s",value);
374:             if (*padoctal)
375:                 fprintf(fpa, "\\207\\ %s\\n",padoctal);
376:             else fprintf(fpa, "\\n");
377:         }
378:     }
379:     else if ((finditem("so=",eptr,value)) != NULL)
380:     {
381:         fprintf(fpa, "          g1 = %s\\200o",value);
382:         if (*padoctal)
383:             fprintf(fpa, "\\207\\ %s\\n",padoctal);
384:         else fprintf(fpa, "\\n");
385:         fprintf(fpa, "          g2 = %s\\200x",value);
386:         if (*padoctal)
387:             fprintf(fpa, "\\207\\ %s\\n",padoctal);
388:         else fprintf(fpa, "\\n");
389:         fprintf(fpa, "          g3 = %s\\200#",value);
390:         if (*padoctal)
391:             fprintf(fpa, "\\207\\ %s\\n",padoctal);
392:         else fprintf(fpa, "\\n");
393:         fprintf(fpa, "          g4 = %s\\200@",value);
394:         if (*padoctal)
395:             fprintf(fpa, "\\207\\ %s\\n",padoctal);
396:         else fprintf(fpa, "\\n");
397:         fprintf(fpa, "          g5 = %s\\200+",value);
398:         if (*padoctal)
399:             fprintf(fpa, "\\207\\ %s\\n",padoctal);
400:         else fprintf(fpa, "\\n");

```

```

401:             fprintf(fpa,"          g6 = %s\\200%",value);
402:             if (*padoctal)
403:                 fprintf(fpa,"\\207\\%s\\n",padoctal);
404:             else    fprintf(fpa,"\\n");
405:             if ((finditem("se=",eptr,value)) != NULL)
406:             {
407:                 fprintf(fpa,"          gf = %s",value);
408:                 if (*padoctal)
409:                     fprintf(fpa,"\\207\\%s\\n",padoctal);
410:                 else    fprintf(fpa,"\\n");
411:             }
412:         }
413: /*
414:  *  initialization strings
415:  */
416:         if ((finditem("is=",eptr,value)) != NULL)
417:         {
418:             fprintf(fpa,"          is = %s",value);
419:             if (*padoctal)
420:                 fprintf(fpa,"\\207\\%s\\n",padoctal);
421:             else    fprintf(fpa,"\\n");
422:         }
423:
424: /*
425:  *  close up and go home
426:  */
427:         fprintf(fpa,"!\\n");
428:         fclose(fpa);
429:     }
430: /*
431:  *  end of main
432:  */
433: /*
434: *****
435:  *
436:  *  find requested string in the string. (first occurrence only)
437:  *
438:  *  'test' is the item to be located
439:  *  'eptr' is the string address to begin searching
440:  *  'value' is the place to put the data following the
441:  *  test item. (up to next ':')
442:  *
443:  *  return NULL if not found
444:  *  return pointer to the item if found
445:  *  return the TERMCAP value string in 'value'
446:  *  if a pad value is present return it in 'padvalue'
447:  *  and the octal equivalent in 'padoctal'
448:  *  if '\\E' or '^x' stuff is present, change it to octal
449:  *
450: *****

```

```

451: */
452: char *
453: finditem(test,eptr,value)
454: char *test, *eptr, *value;
455: {
456:     register char *p1, *p2, *p3;
457:
458:     *padvalue = *padoctal = '\0';
459:     for( ; *eptr ; eptr++ )
460:     {
461:         for(p1=eptr,p2=test,p3=NULL; *p1,*p2; p1++,p2++ )
462:         {
463:             if (*p2 == *p1)
464:                 p3 = p1;
465:             else
466:             {
467:                 p3 = NULL;
468:                 break;
469:             }
470:         }
471:         if (p3)
472:         {
473:             for(p2=value; *p1 && *p1 != ':'; )
474:             {
475:                 *p2++ = *p1++;
476:             }
477:             *p2 = '\0';
478:             translate(value);
479:             return(eptr);
480:         }
481:     }
482:     return(NULL);
483: }
484: /*
485: *****
486: * translate standard syntax items from termcap style to TFI style
487: * replace the input value with the results
488: *****
489: */
490: translate(value)
491: char *value;
492: {
493:     char newvalue[100];
494:     char work[11];
495:     int i;
496:     register char *p1, *p2, *w;
497:
498:     *newvalue = *padvalue = *padoctal = '\0';
499:
500:     /* save any leading decimal digits*/

```

```

501:      /* on the assumption that they are*/
502:      /* padding for time delay */
503:      for(pl=value, p2=padvalue; isdigit(*pl) ; )
504:      {
505:          *p2++ = *pl++;
506:      }
507:      *p2 = '\0';
508:      if(*p2 && *pl == '*')
509:      {
510:          pl++;
511:          *p2++ = '0'; /*weight x 10*/
512:          *p2 = '\0';
513:      }
514:      if (*padvalue)
515:      {
516:          if(i = atoi(padvalue))
517:          {
518:              strcpy(work,"");
519:              w = &work[(strlen(work))];
520:              do
521:              {
522:                  * -- w = od[(int)(i%8)];
523:              } while ((i /= 8) != 0);
524:              strcpy(padoctal,w);
525:          }
526:          else *padoctal = '\0';
527:      }
528:
529:      /* transform escape character "\E" to "\33" */
530:      /* and "^x" control characters to octal*/
531:      for(p2=newvalue; *pl ; )
532:      {
533:          if(*pl == '^')
534:          {
535:              pl++;
536:              *p2++ = '\\';
537:              i = (*pl & 0x1f);
538:              pl++;
539:              strcpy(work,"");
540:              w = &work[(strlen(work))];
541:              do
542:              {
543:                  * -- w = od[(int)(i%8)];
544:              } while ((i /= 8) != 0);
545:              while(*w) *p2++ = *w++ ;
546:          }
547:          else if(*pl == '\\')
548:          {
549:              *p2++ = *pl++;
550:          }

```

```

551:             if(*p1 == 'E')
552:             {
553:                 p1++;
554:                 *p2++ = '3';
555:                 *p2++ = '3';
556:             }
557:         }
558:     else      *p2++ = *p1++;
559: }
560: *p2 = '\0';
561: /*      printf("old>%s< new>%s<\n",value,newvalue); */
562: strcpy(value,newvalue);
563: }
564: /*
565: *****
566: *  special translation routine for cursor movement
567: *****
568: */
569: transxy(value)
570: char *value;
571: {
572:     int flagr;
573:     char valued[10];
574:     char oldvalue[100];
575:     char newvalue[100];
576:     char c;
577:     register char *new, *p1, *p2;
578:
579:     strcpy(oldvalue,value);
580:     /*
581:      * test for reversed sequenced "%r"
582:      * if present, set flag and flush "%r"
583:      */
584:     p1=oldvalue;
585:     flagr = 0;
586:     do
587:     {
588:         if ((p2 = index(p1,'%')) != NULL)
589:         {
590:             new = p2;
591:             new++;
592:             if (*new == 'r')
593:             {
594:                 flagr++;
595:                 p1 = ++new ;
596:                 do
597:                 {
598:                     *p2++ = *p1++;
599:                 } while(*p1) ;
600:                 *p2 = '\0';

```

```

601:                                break;
602:                                }
603:                                else    pl = new;
604:                                }
605:                                else    break ;
606:
607:                                } while(!flagr) ;
608:                                /*
609:                                * test for '%%' sequences and
610:                                * if present, convert to single "%"
611:                                */
612:                                pl=oldvalue;
613:                                while(*pl)
614:                                {
615:                                    if ((p2 = index(pl,'%')) != NULL)
616:                                    {
617:                                        pl = new = ++p2;
618:                                        if (*new == '%')
619:                                        {
620:                                            do
621:                                            {
622:                                                *p2++ = *pl++;
623:                                            } while(*pl) ;
624:                                            *p2 = '\0';
625:                                            pl = new;
626:                                        }
627:                                    }
628:                                    else    break ;
629:                                }
630:                                /*
631:                                * find value to be added to x
632:                                */
633:                                pl=oldvalue;
634:                                new=newvalue;
635:                                if ((p2 = index(pl,'%')) != NULL)
636:                                {
637:                                    if (*(p2 +1) == '+')
638:                                    {
639:                                        for( ; pl < p2 ; *new++ = *pl++) ;
640:                                        pl++; /* flush "%" */
641:                                        pl++; /* flush "+" */
642:                                        *new++ = '\\';
643:                                        *new++ = '2';
644:                                        *new++ = '0';
645:                                        if (flagr)
646:                                            *new++ = '2';
647:                                        else    *new++ = '1';
648:                                        do
649:                                        {
650:                                            *new++ = *pl++;

```

```

651:             *new = '\0';
652:             } while(*pl && (*pl != '%')) ;
653:         }
654:     }
655:     if ((p2 = index(pl,'%')) != NULL)
656:     {
657:         if (*(p2 + 1) == '+')
658:         {
659:             for( ; pl < p2 ; *new++ = *pl++) ;
660:             pl++; /* flush "%" */
661:             pl++; /* flush "+" */
662:             *new++ = '\\';
663:             *new++ = '2';
664:             *new++ = '\0';
665:             if (flagr)
666:                 *new++ = '1';
667:             else *new++ = '2';
668:             do
669:             {
670:                 *new++ = *pl++;
671:                 *new = '\0';
672:             } while(*pl && (*pl != '%')) ;
673:         }
674:     }
675:     for( ; *pl ; *new++ = *pl++) ;
676:     /* printf("old>%s< new>%s<\n",value,newvalue); */
677:     strcpy(value,newvalue);
678: }
679: /*
680: *****
681: * end of source
682: *****
683: */

```

Characters: 15364

**END
DATE
FILMED**

MAR 15, 1984